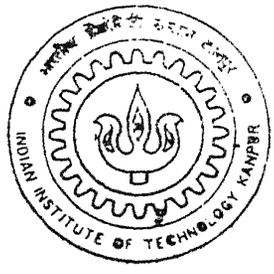
SOME ASPECTS IN CLASSIFICATION OF REMOTELY SENSED IMAGES USING ERROR BACKPROPAGATION ARTIFICIAL NEURAL NETWORK

By

Kumar Ashish Tiwari



DEPARTMENT OF CIVIL ENGINEERING

Indian Institute of Technology Kanpur

APRIL, 2002

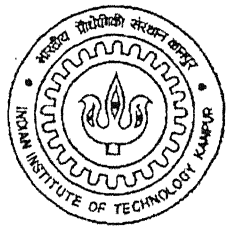
Some Aspects in Classification of Remotely Sensed Images using Error Back-propagation Artificial Neural Network

A thesis submitted
in partial fulfillment of the requirements
for the degree of

Master of Technology

by

Kumar Ashish Tiwari



to the
Department of Civil Engineering

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

April 2002

POST GRADUATE OFFICE
THESIS SUBMITTED
ON 12-4-2002
I.I.T. KANPUR

CERTIFICATE

It is certified that the work presented in this thesis titled "Some Aspects in Classification of Remotely Sensed Images using Error Back-propagation Artificial Neural Network " has been carried out by Kumar Ashish Tiwari (R.N. Y010323) under my supervision and has not been submitted elsewhere for a degree.

Date 11/4/02

Onkar Dikshit

Assoc. Professor
Department of Civil Engineering
IIT Kanpur

4 FEB 2003
पुरुषोत्तम काशीनाथ केलकर पुस्तकालय
भारतीय जैवौषिकी संस्थान कानपुर
अभ्यागत क्र० A 141877

Acknowledgment

I express my deep sense of gratitude to my thesis supervisor, *Dr. Onkar Dikshit* for his insightful teaching of the subject matter. I also thank him for various creative suggestions about the content and organization of my thesis.

I also thank all my teachers at IITK for imparting me very best of education. I also thank *Virendra Pathak* and *S.K.Katiyar*, Ph.D. scholars with the Geoinformatics division, for their continuous encouragement, suggestions and support.

I express thanks to my batch mates *K.M.Reddy*, *Vibhor Jain*, *Harveen Punihani*, *Deepak R. Mishra* and *Jadunandan Dash*, with whom I had great time studying various courses. I also thank my wing mates *Vinod Pathari*, *Prithvi P. Singh Bisht*, *Pijush Ghosh* and *Jayadeep U.B.* for making my stay very pleasant and enjoyable. Thanks are due to my seniors and juniors also for giving me a memorable stay.

I also thank *Mr. Igor Fisher* (*University of Tubingen, Germany*); who is one the members of *JavaNNS* development team, for his help on *JavaNNS* software during the initial stages through e-mail.

Last but not the least, I want to thank *Mr. G.P. Mishra* for his affection, love and caring throughout my stay. Thanks are due to *Ramkishen* also.

Date 11-4-2002
Kanpur

Kumar Ashish Tiwari

Contents

List of Tables

List of Figures

List of Plates

Abstract

Neural network can be successfully used to classify remotely sensed images. Issues like network size, training sample set size and textural information incorporation and their effect on neural network classification accuracy have been investigated in the current study. Results show that there is no significant effect of network size on classification accuracy, although a minimum complexity network is required for classification. A minimal training sample set comprising of $10n$ pixels/class (where n is the number of features) can perform similar to bigger samples. Neural network may give better classification accuracy for spectrally varying classes as compared to GML classifier. Texture information incorporation is done using conventional and window-based approaches. Window based approach is classifying spectrally varying classes in a better manner as compared to conventional approach. This may be because of better neighborhood information incorporation by window-based approach.

1 Introduction

1.1 Background of the work

The remote sensing satellites Landsat, SPOT, ERS, IRS etc. are producing huge amount of data that can be used for various scientific and technological advancements. While such data gives us an opportunity to address many fundamental environmental issues in more depth, it also opens up new challenges for data processing and data interpretation. Traditional parametric statistical approaches to supervised classification like Gaussian maximum likelihood classifier, and minimum distance-to-means, have been successfully used in the past for classifying remotely sensed imagery. These approaches have certain limitations. They depend on the assumption of a multivariate Gaussian (normal) distribution for the data to be classified. Each class in feature space is assumed to have an *n*- dimensional (where *n* is the number of input wavebands) multivariate Gaussian distribution.

The problem with the statistical approach is that the data in the feature space may not follow the assumed model (Atkinson and Tatnall, 1997). Further, it is possible that a single class may be represented at multiple places in the feature space. This is particularly the case when classes are heterogeneous in nature. Consequently, statistical approaches may be seen as restrictive because of the underlying assumption of the model. A further problem with statistical approaches is that they require non-singular (invertible) class specific covariance matrices.

One of the main advantages of neural networks for classification is that they are distribution free, that is, no underlying model is assumed for the multivariate distribution of the class specific data in feature space. It is, therefore, possible for a single class to be represented in feature space as a series of clusters in place of a single cluster. Thus, a fundamental difference between statistical and neural approaches to classification is, that statistical approaches depend on an assumed *model* whereas neural approaches depend on *data*. It is this reason that neural networks are suitable for integrating data from different sources. It is in the above context of these requirements that artificial neural networks are currently being applied in a wide variety of remote sensing applications. The use of artificial neural network for remote sensing data interpretation has been motivated by the realization that the human brain is very efficient at processing vast quantities of data from a variety of different sources. Neurons in the human brain receive inputs from other neurons to produce an output, which is then passed to other neurons. A mathematical model based on the actions of the biological neuron may be implemented to process and interpret different type of data. In this sense, neural networks are an artificial intelligence approach. Neural networks in the simplest sense can be seen as data transformers (Pao, 1989), where the objective is to associate the elements in one set of data with the elements in the second set. When applied to classification, they are concerned with the transformation of data from feature space to class space. Thus, neural networks belong to the same class of techniques as automated pattern recognition. The rapid increase in use of neural networks is due to their ability to (Atkinson and Tatnall, 1997):

(i) perform more accurately than other techniques such as statistical classifiers, particularly when the feature space is complex and the source data has different statistical distributions;

(ii) perform more rapidly than other techniques such as statistical classifiers;

(iii) incorporate a *priori* knowledge and realistic physical constraints into the analysis;

(iv) incorporate different types of data (including those from different sensors) into the analysis, thus facilitating synergistic studies.

1.2 Problem definition

Although, neural networks have been used to classify data accurately, there are problems in their use. One basic issue is the desired size or complexity of the network (Foody and Arora, 1997). This is effectively a function of the number of weighted connections in the network, which in turn is a function of the number of network layers and the number of units in each of these units. As opposed to a simple network, a complex network is more able to achieve an accurate characterization of the training pixels, but have a lower capacity to generalize and correctly classify testing pixels. Other issues like number of training iterations may also have an effect on the learning and generalization ability of the network. Since the number of input and output units are effectively fixed by design, the problem in defining architecture relates to the nature of hidden layer(s). Specifically, the researcher has to find out how many hidden layers and units to use in defining the network architecture.

The characteristics of the data used to train a supervised classification have a considerable influence on the quality of the resulting classification (Campbell, 1981). It is essential that the training data provide a representative description of each class. A requirement is that the size of training set be at least 10 to 30 times the number of bands (Mather, 1987; Piper, 1992). The required training size may therefore be large. The number could further rise because of other issues (*e.g.* the effect of background soil on the spectral response of crop). Thus, acquiring such training sets is difficult for classification involving a large number of classes. Consequently, many investigations have been based on sample sizes below the generally accepted guidelines and thus may not have fully exploited the information content of the remotely sensed data. The lack of distributional assumption makes artificial neural network an attractive alternative to the conventional statistical classification schemes. It has been proposed that artificial neural network classification may be performed on smaller even minimal training sets (Hepner *et al.*, 1990, Foody *et al.*, 1995).

Neighborhood information can be incorporated in form of context and texture. When humans visually interpret remotely sensed imagery, they synergistically take into account, in addition to spectral value, context, edges and texture. Conversely, most digital image processing classification algorithms are based only on the use of spectral (tonal) information. Thus, it is not very surprising that there have been efforts to incorporate some of these other characteristics into digital classification procedure. A discrete tonal feature is a connected set of pixels that have the same or almost the same gray shade. When a small area of image (e.g., a 3x3 area) has little variation of discrete tonal features, the dominant property of that area is a gray shade (Jensen, 1996). Conversely, when a small area has a wide variation of discrete tonal features, the dominant

property of that area is texture. Most researchers trying to incorporate texture into the classification process have attempted to create a new texture image that can then be used as another feature or band in the classification. Thus, each new pixel of the texture image has a brightness value that represents the texture at that location. Another alternative (Bischof *et al.*, 1992) is presenting the network with the entire window of pixels normally used to calculate a local texture value. It allows the network to infer an arbitrary texture as necessary to help separate the classes. In the current study, the second (presenting the network with window of pixels) approach has been investigated.

1.3 Objectives and scope of the work

The following are the specific objectives of the present study

1. To compare the classification performance of back-propagation artificial neural network and Gaussian maximum likelihood classifiers;
2. To evaluate the performance of back-propagation artificial neural network classification while using,

(a) different number of hidden nodes;

(b) different training set size;

3. To evaluate the performance of back-propagation ANN using combined neighborhood (texture) and spectral information.

1.4 Overview of data used and study area

In this work, data from two different sites have been taken for the purpose of analysis. They are cities of *Lucknow* and *Bhopal*, the capitals of two north Indian

states, and adjoining areas. The location of these cities is shown in the map of India in Figure 1.1.

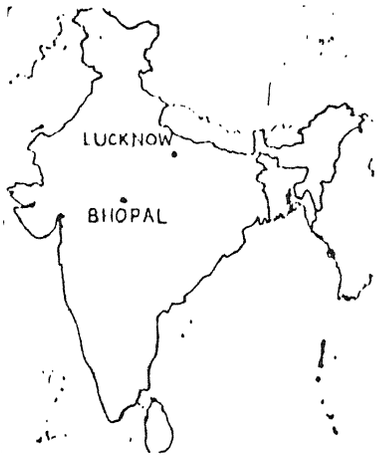


Figure 1.1 Locations of the study sites

The details of the data products for study sites are given in Table 1.1. The data used for *Lucknow* site is IRS 1C, LISS III data comprising of four multi spectral bands taken in Nov 1996. The area covered is predominantly urban and agricultural in nature. It is covered between Latitudes 26°49′ N and 26°57′ N and Longitudes 80°55′ E and 81°01′ E. River *Gomati* passes through the middle left portion of the region. The region to the south of the river is dominantly urban while towards north agriculture and medium urban region is present. In the northern region a reserved forest (*Kukrel*) is clearly identifiable. The data used for *Bhopal* is IRS 1B, LISS II data comprising of four multi spectral bands. The image was taken in Jan 1995. The area is located between Latitudes 23°12′ N and 23°17′N and Longitudes 77°11′E and 77°25′E. The area is urban and a lot of vegetation is also present. In the central part the big *Bhopal* lake can be identified. The false color composites of study sites using NIR, red and green bands are shown in Plates 1.1 and 1.2.

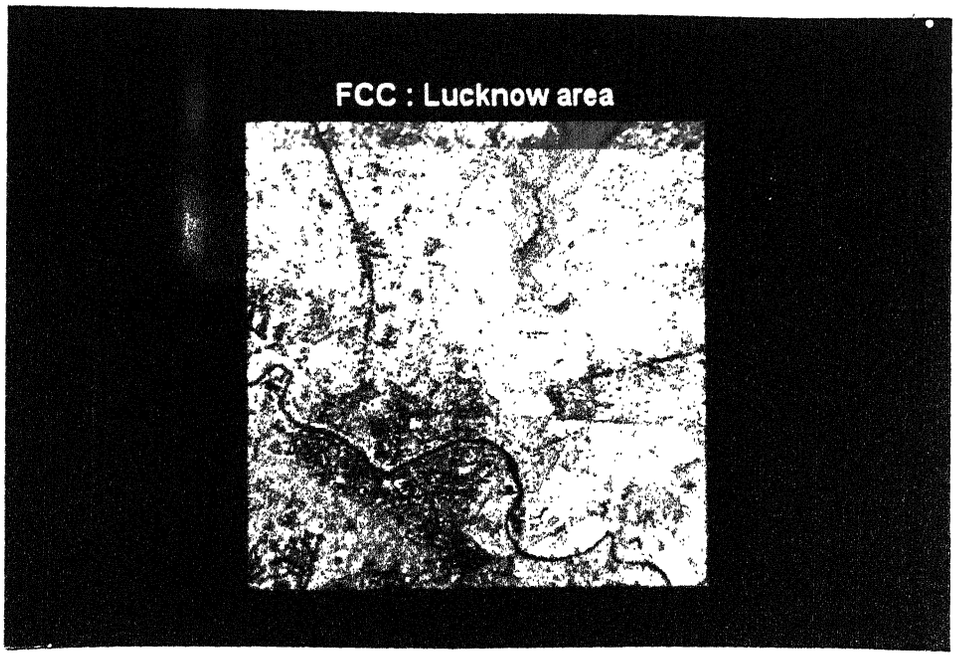


Plate 1.1 False color composite (FCC) of *Lucknow* area generated using NIR, red and green bands.

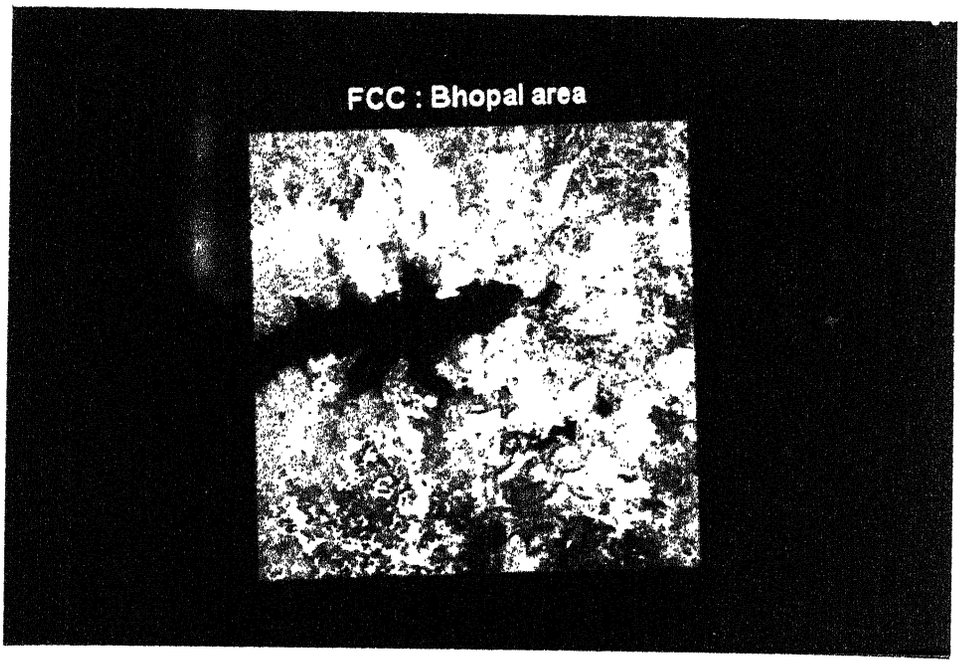


Plate 1.2 False color composite (FCC) of *Bhopal* area generated using NIR, red and green bands.

.

Table 1.1 Satellite data characteristics for study areas.

Satellite /Sensor	Bands	Resolution (m)	Size (pixels)	Wavelength (μ m)	Spectral Region	Path/Row
Lucknow						
IRS 1C LISS III	Band 1	23.5	512 x 512	0.52-0.59	Green	100/052
	Band 2	23.5	512 x 512	0.62-0.68	Red	
	Band 3	23.5	512 x 512	0.77-0.86	NIR	
	Band 4	70.5*	512 x 512	1.55-1.70	SWIR	
Bhopal						
IRS 1B LISS II	Band 1	36.5	512 x 512	0.45-0.52	Blue	51/28
	Band 2	36.5	512 x 512	0.52-0.59	Green	
	Band 3	36.5	512 x 512	0.62-0.68	Red	
	Band 4	36.5	512 x 512	0.77-0.86	NIR	

* resampled to 23.5 m.

1.5 Software and hardware details.

Extensive use of existing software has been done. The main names and their uses are given below:

1. **ILWIS** (Integrated Land and Water Information System, ITC Netherlands): for the making of false color composite images and sample set selection and visualization of classified images.
2. **JavaNNS** (Java Neural Network Simulator): To make several networks, perform training operations on them and for classifying imagery by the trained network.
3. **Microsoft Visual C++ 6.0**: For creating and running various programs in C.

The standard hardware configuration of the PC (known as GIS 1) include Cyrix M II /IBM 6x86MX -233 CPU, DAEWOO 518X color monitor, 20 GB H.D.D. and 160.0 MB RAM.

1.6 Organization of work

The thesis has been organized in 6 chapters. The current chapter provides a brief introduction to the problem, data sources and software and hardware details. Chapter 2 presents literature review, which comprises of introduction to basic concepts in classification, texture, neural network and a survey of the work done by previous investigators in the field. Chapter 3 presents theoretical background for the error back-propagation training in neural networks. Chapter 4 describes methodology for classifying images using neural network. Chapter 5 presents results obtained from the study and discussion of the results, while chapter 6 gives a summary of conclusions.

2 Literature Review

2.0 Introduction

In this chapter, concepts related to classification, Gaussian maximum-likelihood (GML) classification and texture are introduced. They are followed by a brief description about biological neuron and its equivalent mathematical representation, the artificial neuron. After this a simple feed forward network has been described. They are followed by assessment of optimum hidden layer size and optimum sample set size for the network. The next section describes accuracy assessment and various statistics calculation for any classification. Lastly, work by recent investigators in the field of back-propagation artificial neural network has been examined.

2.1 Classification or Pattern recognition

Remotely sensed data can be used to extract useful information about the area to which the data belongs, which is thematic in nature. This process of converting data into useful information, which is of fundamental importance in remote sensing, is called *Pattern Recognition* or simply *Classification.* Before we go forward we can have a look at the terms pattern and class. In multi spectral classification (which means we are using remotely sensed data that has been obtained using more than one band of electromagnetic radiation), a set of values for a single pixel on each of a number of spectral bands may be referred to as a pattern. The variables that define the basis of such a measurement (e.g. values in different bands) are called features. Thus, we can simply say that a

pattern is a measurement of values over a chosen set of features. When we talk about class, we are generally referring to information class as against spectral class. Information class refers to actual earth surface cover types while spectral class is a grouping of pixels of similar spectral response. In remote sensing applications we are more concerned about information classes because it provides information that can be directly used. The way to achieve such a classification is through *supervised classification.*

In a supervised classification, the identity and location of some of the land cover types, such as urban, water and agriculture, are known a priori, through several means other than classification, like field work, aerial surveys etc. Specific sites can be located corresponding to those cover classes on the image. These sites are called training sites. Once these training sites corresponding to different classes have been located their properties (spectral values) can be used to classify the whole image into those classes. This process of assigning a predetermined label on each of the pixels in the image is known as supervised classification. There are several methods to do this. The choice of particular classifier or decision process depends upon the nature of input data and the desired output. Parametric classification algorithms assume that the measurement vectors corresponding to any training class are Gaussian in nature i.e. they follow normal distribution, which is a statistical assumption. The classifiers designed on this assumption are sometimes called as statistical classifiers also. The ones that come under this category are the traditional classification schemes like parallelepiped, minimum-distance-to-means and maximum likelihood classifiers.

However, the assumption of normality may not be valid in several of practical applications, in which classes may be skewed, have a double peak (in cases of heterogeneous classes) etc. and in such cases the use of parametric classifiers is essentially flawed because the assumption for classification schemes are not valid. A recent advancement in the field of supervised classification is using Artificial Neural Network, which is a non-parametric approach. A non-parametric approach means that it does not have any inherent statistical assumption.

2.2 Gaussian maximum likelihood classifier

The maximum likelihood classification rule assigns each pixel having pattern measurements or features X to the class c whose units are most probable or likely to have given rise to feature vector (Jensen, 1996) It assumes that the training data statistics for each class in each band are normally distributed. This classification makes use of the mean measurement vector M_c for each class and the covariance matrix of class c, V_c. The decision rule applied to the unknown measurement vector X is

Decide X is in class c, iff

$p_c \geq p_i$, where $i = 1, 2, 3, \ldots$ all possible classes (2.1)

and

$$p_c = \{-0.5 \log_e[\det(V_c)]\} - [0.5(X - M_c)^T V_c^{-1}(X - M_c)] \quad (2.2)$$

where det (V_c) *is* the determinant of the covariance matrix V_c and X is the measurement vector to be classified.

2.3 Texture

Texture is generally understood as neighborhood property, although it is difficult to define texture. At a simple level texture can be thought of as the variability in tone within a neighborhood, or the pattern of spatial relationships among the gray levels of neighboring pixels (Shih and Schowengerdt, 1983) and is sometimes described in terms such as "rough" or "smooth". Qualitatively texture is defined by various terms like *coarseness, contrast, directionality, likeliness, regularity* and *roughness* (Tamura *et al.*, 1978). *Coarseness* depends upon the pattern and size of texture elements. *Contrast* depends upon the dynamic gray level, polarization of the distribution of black and white on a gray level histogram. *Directionality* depends upon the shape of the elements and placement rules. *Likeliness* refers to the shape of the primitives of texture and supplements *coarseness, contrast* and *directionality. Regularity* relates to the variation in placement rules of the texture elements. *Roughness* is a mental concept related to real world 3-D objects.

In addition to these, texture is essentially a resolution dependent phenomenon. It implies that a change in the scale of imagery will change the whole perception of texture. With the above understanding in mind some definitions of texture can be given (Dikshit, 1992)

1. Those relationships between gray levels in the neighboring resolution cells which contribute to the overall appearance or visual characteristics of an image and are retained even, for example, when a colored scene is transformed into varying levels of gray, are collectively known as the texture of the image.

2. Texture could be defined as a structure composed of a large number of more or less ordered similar elements or patterns without one of these drawing special attention so that the global unitary impression is offered to the observer (Gool *et al.*, 1985)

There are two approaches to texture. In first approach, one-dimensional Grey Level Difference Histogram (GLDH) is calculated. Then using it, texture properties like, e.g. angular second moment, contrast, mean and entropy etc. are calculated and reported as texture. The angular second moment (*asm*) is given as $\sum_{i=1}^{N} p(i)^2$, where $p(i)$ is the normalized values at the ith gray level in GLDH and N is the quantization level. In second approach, a two dimensional gray-tone spatial dependency matrix (also called gray-level co-occurrence matrix, GLCM) is calculated. Haralick *et al.* (1979) proposed gray-tone spatial dependency matrix, which represents the distance and angular spatial relationships over an image sub-region of specified size. Each element of the gray-tone spatial dependency matrix is a measure of the probability of occurrence of two grayscale values separated by a given distance in a given direction. The number of adjacent pixels with gray levels i and j is counted and is placed in element (i, j) of the gray-tone spatial dependency matrix. Four definitions of adjacency are used: Horizontal (0°), vertical (90°), diagonal (bottom left to top right, 45°) and diagonal (top left to bottom right, 135°). Thus four gray-tone spatial dependency matrices can be calculated. The average of these four measures is normally output as the texture value for the pixel under consideration. Haralick proposed 32 texture features to be derived from each of the four matrices. Three of the more widely used include the angular second moment (ASM), contrast (CON) and correlation (COR). They differ

slightly in their definition. Shaban (1999) has reported similar results for both GLDH and GLCM approaches, however GLDH approach is computationally less expensive.

In the present project, a window based approach (Bischof *et al.*, 1992) for implementation of texture has been adopted. In this approach, in place of calculating one single texture value from the GLCM, we give the whole neighborhood information as spectral values to the neural network. This enables the neural network to infer texture on its own.

2.4 Artificial neuron

An artificial neuron is the fundamental processing unit of any artificial neural network. It is the mathematical representation of the biological neuron. The biological neuron is briefly described in the next section.

2.4.1 The biological neuron

The biological neuron is shown in Figure 2.1 with its various connections with other neurons. A typical cell has three major regions: the cell body, which is also called *soma*, the *axon*, and the *dendrites*. Dendrites form a dendritic tree, which is a very fine bush of thin fibers around the neurons body. Dendrites receive information from neurons through axons--long fibers that serve as transmission lines. An axon is a long cylindrical connection that carries impulses from the neuron. The end part of an axon splits into a fine arborization. Each branch of it terminates in a small endbulb almost touching the dendrites of neighboring neurons. The axon-dendrite contact organ is

called a *synapse*. Synapse is where the neuron introduces its signal to the neighboring neuron.

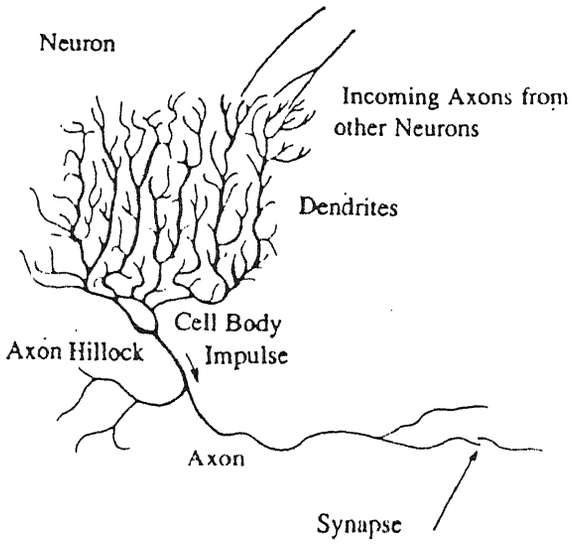


Figure 2.1 The biological neuron (Zurada, 1999)

2.4.2 The mathematical equivalent model

A mathematical equivalent of the biological neuron is discussed in this section. A simple schematic diagram describing a mathematical model is shown in Figure 2.2. In mathematical model, any neuron-processing unit (equivalent to biological processing unit *soma*) has several input activations, which are equivalent to *dendrites* in biological neuron. The relative importance of those connections with respect to the neuron is manifested in the connecting weights between them. The weighted sum of all the inputs is calculated and depending on the sum an output response is calculated. This output is fed to neuron in the next layer as input (similar to the functioning of *axon* in biological neuron).

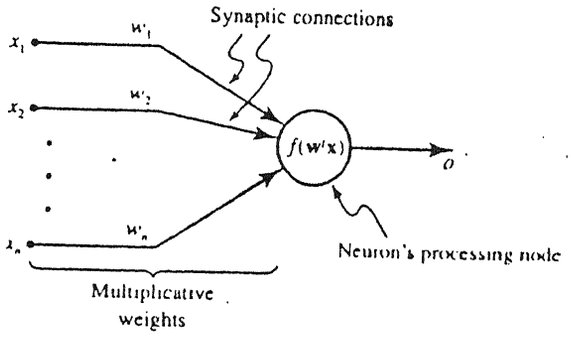


Figure 2.2 A corresponding mathematical model of neuron *i* (Zurada, 1999)

The neuron output signal is given by the following relationship:

$$o = f(\sum_{j=1}^{n} w_{ij} x_j); \tag{2.3}$$

where w_{ij} is the weight corresponding to connection of the ith neuron to the jth signal. x_j is the intensity of jth signal. Subsequently an activation function is imposed upon the sum. A typical activation function is given as

$$f(x) = \frac{2}{1+\exp(-\lambda x)} - 1 \tag{2.4a}$$

where $\lambda > 0$. This is known as general *bipolar activation function* and is a continuous function. As $\lambda \to \infty$, the limit of this function becomes the *binary bipolar activation function*, which is equivalent to *sgn*(x) function (+1 for all $x>0$ and -1 for all $x < 0$). Shifting and scaling the *bipolar activation functions* can obtain *unipolar activation functions*, given as

$$f(x) = 1/(1+\exp(-\lambda x)); \tag{2.4b}$$

As $\lambda \to \infty$, the limit of this function becomes the *binary unipolar activation function* (+1 for all $x>0$ and 0 for all $x < 0$). The *unipolar function* scales the output between 0 and 1. Various activation functions are shown in Figure 2.3.

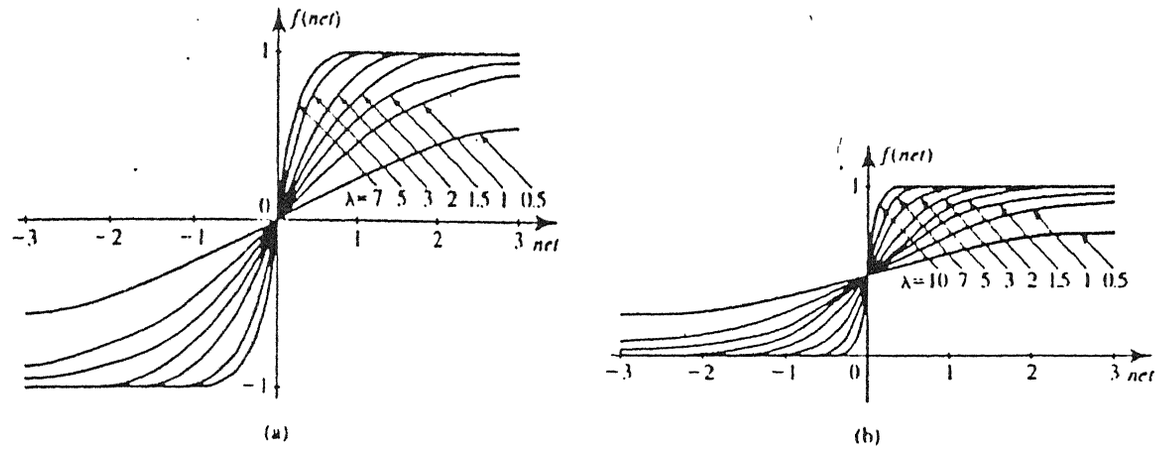


Figure 2.3 (a) Bipolar and (b) unipolar activation functions. (Zurada, 1999)

2.5 Artificial neural network

Artificial neural network is an interconnected network of artificial neurons. With the understanding that a neuron has a number of inputs and one output, we can construct connected networks made of these neurons. These neurons can be connected in a number of ways. The most popular is the feedforward network (Zurada, 1999). A multi layer feedforward network (Figure 2.4) consists of several layers of connected neurons. Any layer can have any number of neurons. Initial excitations are input to all the neurons in the first layer and each neuron produces just one output. These outputs from all the neurons are fed as input excitations to all the neurons in next layer and so on, until a final output is achieved.

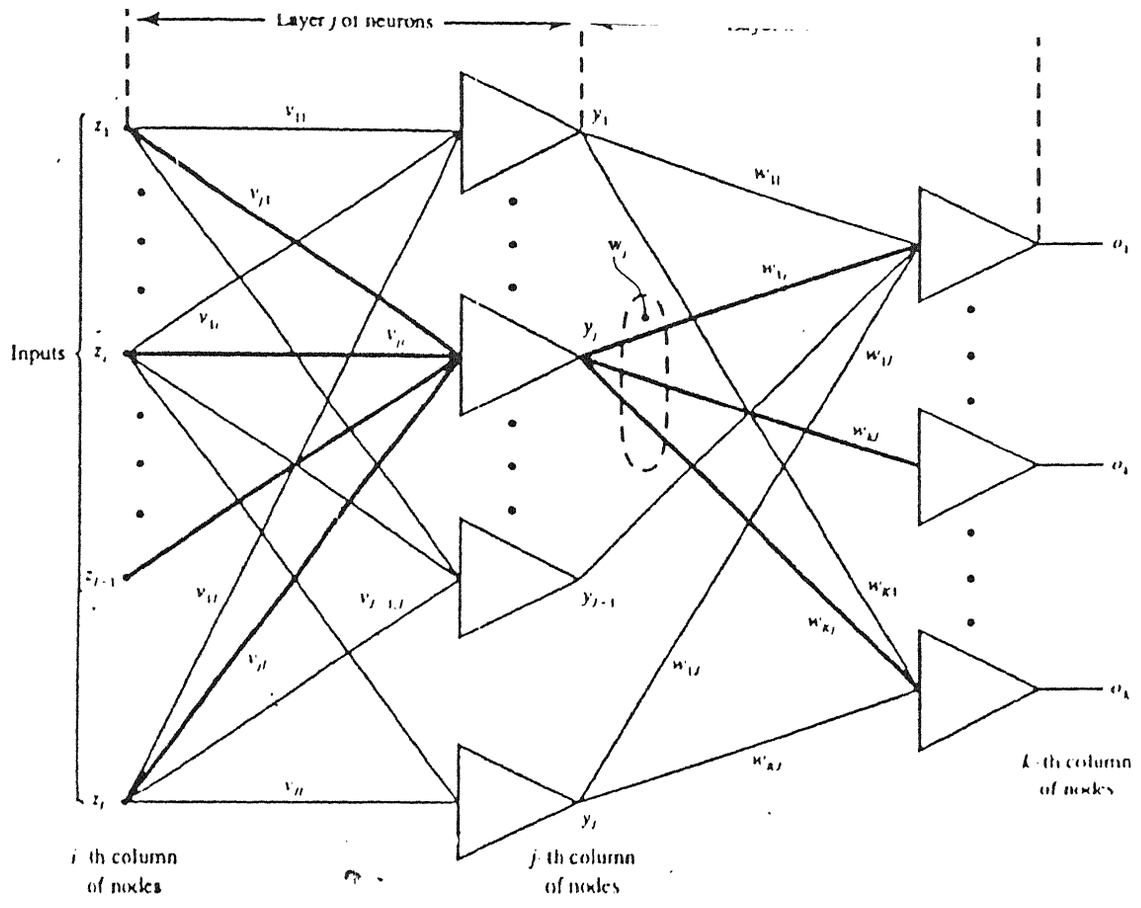


Figure 2.4 A layered feed forward network with three layers (Zurada, 1999)

2.6 Hidden layer size assessment

In defining the network parameters, the input layer size is equal to the number of inputs (bands) and output layer size is equal to the number of classes. With these values set, the remaining parameters are the number of hidden layers and number of nodes per hidden layer. If we assume a three-layer network, the sole parameter to be considered is the number of nodes in the single hidden layer. If we view both neural network and maximum-likelihood classifier as decision making functions defined by a number of parameters (degrees of freedom), then it is logical to use the same number of parameters in each for comparison (Paola and Schowengerdt, 1995). For a three-layer network, assuming one input node per image band and one output node per class, the number of parameters as a function of network structure is

N_{net} = 3 + # weights

= 3 + hidden layer nodes*(bands+classes) (2.5a)

where the first term (3) is the number of parameters needed to specify the size of each of the three layers. For the maximum-likelihood classifier, the corresponding number of parameters is

N_{ML} = 2 + # means + # unique covariances

= 2 + classes* bands+ 1/2*classes*(bands^2+bands) (2.5b)

where 2 is the number of parameters required to specify features and classes. Setting N_{net} equal to N_{ML} and solving for the number of hidden layer nodes, we get

# of hidden layer nodes =

(2+classes*bands+0.5*classes*(bands^2+bands)-3)/(bands+classes) (2.6)

2.7 Optimum sample size assessment

For classification and latter for accuracy assessment the sample set is an important parameter. The question remains that how many minimum (or optimum) number of pixels are required per class so that the analysis performed is statistically valid. There are two schemes for this (Congalton and Green, 1999).

1. Binomial Distribution: The binomial distribution or the normal approximation to the binomial distribution is appropriate for computing the sample size for determining overall accuracy or the accuracy of an individual category. The equations are based on the proportion of correctly classified samples and on some allowable error.
2. Multinomial Distribution: In the case of an error matrix, it is not simply a question of correct or incorrect. Instead, it is a matter of which error or which categories are being confused. Given an error matrix with n land cover categories, for a given category there is one correct answer and $n-1$ incorrect answers. Therefore, the use of binomial distribution for determining the sample size for an error matrix is not appropriate. Instead, the use of multinomial distribution is suggested.

Next, in order to determine the required sample size, the precision b_i for each parameter in the multinomial population must be specified. Generally for assessing the accuracy of remotely sensed data, an absolute precision is set for the entire classification and not for each category or each cell, therefore $b_i = b$. If there is no *a priory* knowledge about the values of probability of occurrence of each of the classes in the image, we can assume a worst probability as equal to 1/2. In this case the sample size N required to generate a valid error matrix is given as

$$N = B/4b^2. \tag{2.7}$$

where B is the upper α/k *x100th* percentile of the χ^2 distribution with on degree of freedom and b is the absolute precision. For 95% confidence, from the χ^2 - table, we can obtain the value of B as 7.879 while for 99% confidence the value is 10.825.

2.8 Accuracy assessment

We can assume that n samples are distributed into k^2 cells where each sample is assigned to one of k categories in the remotely sensed classification. Let n_{ij} denote the number of samples classified into category i in the remotely sensed classification and in category j in the reference data set. Letting

$$n_{i+} = \sum_{j=1}^{k} n_{ij} \tag{2.8}$$

be the number of samples classified into category i in the remotely sensed classification and

$$n_{+j} = \sum_{i=1}^{k} n_{ij} \tag{2.9}$$

be the number of samples classified into category j in the reference data set. Overall accuracy between remotely sensed classification and the reference data can be computed as follows:

$$\text{Overall accuracy} = \sum_{i=1}^{k} n_{ii} \Big/ n \tag{2.10}$$

Producer's accuracy (error of omission) for class j can be computed as equal to

$$n_{jj} \Big/ n \tag{2.11}$$

User's accuracy (error of commission) for class i can be computed as equal to

$$n_{ii}/n \tag{2.12}$$

Error of commission and error of omission are two measures of accuracy, so a measure of their agreement of accuracy is proposed as kappa. The kappa analysis is a discrete multivariate technique used in accuracy assessment, which removes chance agreements. The result of this analysis is the K_{hat} values which truly represent the classification accuracy of any class as well as overall after removing chance match.

The overall kappa, κ for classification is given as

$$\kappa = \frac{n\sum_{i=1}^{k} n_{ii} - \sum_{i=1}^{k} n_{i+}n_{+i}}{n^2 - \sum_{i=1}^{k} n_{i+}n_{+i}} \tag{2.13}$$

And the approximate large sample variance σ^2 of the overall classification is given as

$$\sigma^2(\kappa) = \frac{1}{n}\left\{\frac{t_1(1-t_1)}{(1-t_2)^2} + \frac{2(1-t_1)(2t_1t_2-t_3)}{(1-t_2)^3} + \frac{(1-t_1)^2(t_4-4t_2^2)}{(1-t_2)^4}\right\} \tag{2.14}$$

where

$$t_1 = \frac{1}{n}\sum_{i=1}^{k} n_{ii}$$

$$t_2 = \frac{1}{n^2}\sum_{i=1}^{k} n_{i+}n_{+i}$$

$$t_3 = \frac{1}{n^2}\sum_{i=1}^{k} n_{ii}(n_{i+} + n_{+i})$$

$$t_4 = \frac{1}{n^3}\sum_{i=1}^{k}\sum_{j=1}^{k} n_{ij}(n_{j+} + n_{+i})^2.$$

The kappa, κ_i and large sample variance for kappa, $\sigma^2(\kappa_i)$ for any individual class can be calculated using the following formulae

$$\kappa_i = \frac{nn_{ii} - n_{i+}n_{+i}}{nn_{i+} - n_{i+}n_{+i}}, \tag{2.15}$$

$$\sigma^2(\kappa_i) = \frac{n(n_{i+} - n_{ii})}{[n_{i+}(n - n_{+i})]^3}[(n_{i+} - n_{ii})(n_{i+}n_{+i} - nn_{ii}) + nn_{ii}(n - n_{i+} - n_{+i} + n_{ii})] \tag{2.16}$$

The test statistics, whether two classification schemes are significantly different from each other, is expressed by

$$Z_{ij} = \frac{|\kappa_i - \kappa_j|}{\sqrt{\sigma^2(\kappa_i) + \sigma^2(\kappa_j)}} \tag{2.17}$$

At 95% confidence level the critical value is 1.96 *i.e.* two error matrices are significantly different with 95% confidence if Z statistics value for them is greater than 1.96. If Z statistics differs by more than 3.0, confidence level goes up to 99%.

2.9 Work by previous researchers

Kanellopoulous et al. (1991) used a two level hierarchical network to classify land cover classes because, they argued, most land cover schemes are hierarchical in nature. They reported little improvement in accuracy. Lio and Xiao (1991) proposed block back-propagation or selective connection networks but they didn't compare it with the corresponding fully connected network.

Heerman and Khazenie (1992) used a variety of training set sizes to conclude that accuracy didn't improve significantly for larger training sets. They devoted a section of their paper to the preparation of their training set. They refer to a 'picking and packing' technique. The 'picking' is accomplished by first employing an unsupervised clustering algorithm. Then small homogeneous regions are hand selected to represent the ground cover classes. Eliminating duplicate pixels within a class avoids redundant calculations and insures that contradictory input information is not presented to the network, but it curtails the networks ability to learn generalized heterogeneous input.

Paola and Schowengerdt (1995) used Landsat Thematic Mapper images of Tucson, AZ and Oakland, California to do a detailed comparison of the back-propagation neural network and maximum-likelihood classifiers for urban land use classification. They concluded that neural network is more robust an approach for heterogeneous classes, because it is nonparametric in nature as against conventional approaches.

Foody and Yates (1995) examined the effect of training size and composition on artificial neural network classification. They observed that in the classification of the remotely sensed data set the classification accuracy was increasing significantly as a result of increasing the number of training cases for abundant classes in the image. They also warned towards careful selection of training set for the application in hand.

Foody (1995) examined neural network for its potential for soft classification. He tried to associate neural network class output value for a particular class of a mixed class pixel to the percentage of area that particular class in that pixel represents actually on ground. He found that the output values in themselves were not strongly correlated to the pixel compositions.

In a review article, Paola and Schowengerdt (1995) analyzed five aspects of neural network classification namely input encoding, output encoding, architecture, training algorithms and comparison to conventional classifiers. They suggested the presenting of entire window of pixels normally used to calculate texture, which allows the network to infer an arbitrary texture as necessary to separate the classes. The pixel values should be normalized between 0 and 1, which avoids the use of scale vector each time sigmoid function is evaluated. Using raw values may lead to stalling of network learning due to early saturation (Kanellopoulos and Wilkinson, 1997). They suggest that generally for multi-spectral imagery a three layer (single hidden layer) fully interconnected network is sufficient and a four-layer network is unnecessary.

Foody and Arora (1997) studied the affect of four factors on neural network classification accuracy, namely network architecture, training set size,

discriminating variables and testing data characteristics. They found that the accuracy of neural network classification tends to increase with training set size although the increase in accuracy is not significant after a certain training set size. The number of discriminating variables had a positive correlation with accuracy i.e. the accuracy was maximum when all the spectral bands available were used for analysis. Also, classification accuracy increased with the increase in testing set size. They also observed that neural network architecture has no significant effect on classification accuracy.

As far as comparison with conventional classifiers (like GML) is concerned, most of the authors (Blonda *et al.*, 1993; Fierens *et al.*, 1994; Paola and Schowengerdt, 1995; Yoshida and Omatu, 1994; Kanellopoulos *et al.*, 1993; Li *et al.*, 1993; Bischof *et al.*, 1992; Heermann and Khazenie, 1992; Kiang, 1992; Liu and Xiao, 1991) have reported similar or superior classification accuracy.

In a paper suggesting strategies and best practice for neural network image classification Kanellopoulos and Wilkinson (1997) suggested a variety of measures. They suggested normalization of spectral values, avoiding "butterfly effect"(drastic change in output value for a small change in input value), use of conjugate descent algorithm and use of combined neural and non-neural methods for accuracy improvement.

Summary of various data encoding and network structures used by several authors is described in Table 2.1.

Table 2.1 Summary of various data encoding and network structures (Paola and Schowengerdt, 1995)

Author	Imagery	Input data encoding	Network structure
Benediktsson *et al* (1990 a)	MSS, elevation slope, aspect data	Grey coding	56-32-10 56-32-4
Benediktsson *et al.* (1990 b)	60 bands of Simulated HIRIS	Binary coding 12 bits per band	240-15-3,480-15-3 720-20-3
Hepner *et al.* (1990)	4 TM bands	3x3 window in each band	36-10-4
Key *et al.* (1990)	Merged AVHRR and SMMR	Individual pixel values	7-10-12
Kannellopoulos *et al.* (1991)	2 date SPOT	Individual pixel values	6-18-54-20
Heermann Khazenie (1992)	3 TM bands	Binary data, 8 bits per band	24-24-5
Bischof *et al.* (1992)	7 TM bands	Coarse coding, also with 5x5 window	91-5-4, 116-8-4 140-8-4
Li and Si (1992)	10 band airborne spectrometer	Individual pixel values	10-7-3
Wilkinson *et al.* (1992)	2 dates of 3 SPOT bands	Individual pixel values	3-15-7 6-21-7
Civco (1993)	6 TM bands	Individual pixel values	6-15-15
Dreyer (1993)	3 SPOT bands, texture calculations	Individual pixel values	3-13-12-9,6-8-8-9, 42-7-7-9

The work done by various investigators in the field of back-propagation artificial neural network is summarized in Table 2.2

Table 2.2 Summary of the work done by recent investigators in the field of back-propagation ANN.

Author	Work
Kanellopoulous *et al.* (1991)	Two level hierarchical network for classification
Heerman & Khazenie (1992)	Effect of training set size on accuracy
Bischof (1992)	Use of window method as a texture measure
Foody (1995)	Soft classification of the neural network output
Paola & Schowengerdt (1995)	Comparison of back-propagation neural network and maximum likelihood classifier for urban environment
Foody & Yates (1995)	Effect of training set size and composition on neural network accuracy
Foody and Arora (1997)	Effect of network architecture, training set size, discriminating variables and testing data characteristics on ANN classification accuracy.

3 Theory of back-propagation ANN

3.0 Introduction

In this chapter, the basic theory underlying back-propagation artificial neural network has been described. The chapter begins with a description of concept of supervised and unsupervised learning in neural network. After that a brief description of various ANN learning rules is given. That is followed by concepts of a multi-layer network and generalized delta-learning rule. Lastly, an algorithm for error back-propagation training has been given. Zurada (1999) has been taken as reference for the theory of artificial neural network.

3.1 Supervised and unsupervised learning

By learning, we mean a process of forcing a network to yield a particular response to a specific input. The concept of feedback plays a central role in learning. There are two types of learning: learning with supervision and learning without supervision. In supervised learning we assume that at each instant of time when the input is applied, the desired response **d** of the system is provided by the teacher. The distance **ρ[d,o]** between the actual and desired response serves as an error measure and is used to correct network parameters externally. Since adjustable weights are assumed, the teacher may implement a reward and punishment scheme to adapt the network's weight matrix **w**. Typically, supervised learning rewards accurate classifications or associations and punishes those which yield inaccurate responses. The teacher estimates the negative error gradient direction and reduces the error accordingly. Most

supervised learning algorithms reduce the stochastic minimization of error in multi-dimensional weight space.

In learning without supervision, the desired response is not known; thus explicit error information cannot be used to improve network behavior. In this mode of learning, the network must discover for itself any possibly existing patterns, regularities, separating properties etc. Since no information is available as to correctness or incorrectness of responses, learning must be accomplished based on observations of responses to inputs. Thus, the technique of unsupervised learning is often used to perform clustering as the unsupervised classification of the objects without providing the information about the actual class.

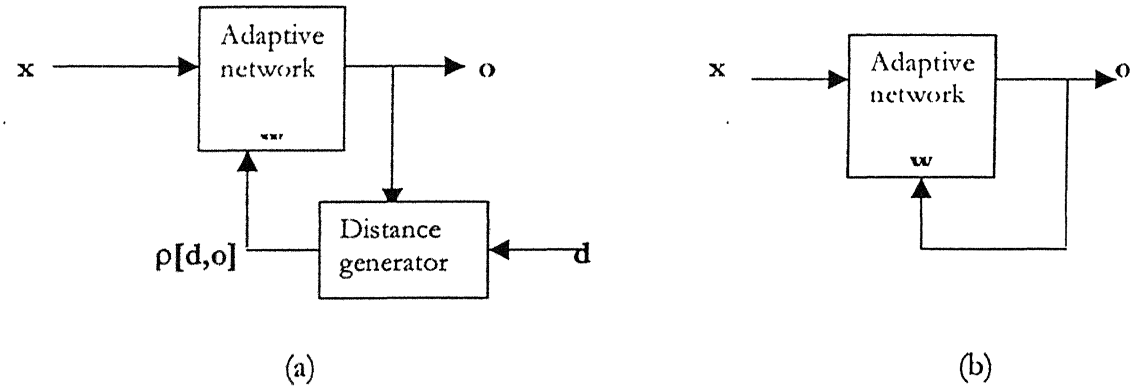


Figure 3.1 Block diagrams for (a) supervised and (b) unsupervised learning.

3.2 Neural network learning rules

Let the weight $\mathbf{w}_i$, or its components w_{ij} connecting the jth input with the ith neuron. In general the jth input can be an output of another neuron or it can be an external input. The following general learning rule is adopted in neural network studies (Amari, 1990): *The weight vector* $\mathbf{w}_i=[w_{i1}\ w_{i2}\ \ldots\ w_{in}]^t$ *increases in proportion to the product of input* x *and learning signal* $\mathbf{r}$. The learning signal $\mathbf{r}$ is generally a function of $\mathbf{w}_i$, $\mathbf{x}$ and teacher's signal $\mathbf{d}_i$. Thus, the increment of the weight vector $\mathbf{w}_i$ produced by the learning step at time t is

$$\Delta\mathbf{w}_i = c\,\mathbf{r}[\mathbf{w}_i,\mathbf{x},\mathbf{d}_i]\,\mathbf{x} \tag{3.1}$$

where c is a positive number called the learning constant.
Some important learning rules are listed as below :

(a) Hebbian learning rule: It is suitable for only for unsupervised learning. For the Hebbian learning rule the learning signal is equal simply to the neuron's output. Thus, we have

$$\mathbf{r}=f(\mathbf{w}_i^t\mathbf{x}) \tag{3.1.1a}$$

and the weight increment becomes

$$\Delta\mathbf{w}_i = c\,f(\mathbf{w}_i^t\mathbf{x})\mathbf{x} \tag{3.1.1b}$$

(b) Perceptron learning rule: For the Perceptron learning rule the learning signal is equal to difference in the neuron's output and desired response. Thus, we have

$$\mathbf{r}=\mathbf{d}_i - f(\mathbf{w}_i^t\mathbf{x}) \tag{3.1.2a}$$

and the weight increment becomes

$$\Delta\mathbf{w}_i = c\,(\mathbf{d}_i - f(\mathbf{w}_i^t\mathbf{x}))\mathbf{x} \tag{3.1.2b}$$

(c) Delta learning rule: The learning signal for this rule is defined as

$$r=[d_i-f(w_i^t x)]f'(w_i^t x) \qquad (3.1.3)$$

The term $f'(w_i^t x)$ is the derivative of the activation fuction $f(x)$. The delta-learning rule is only valid for continuous activation functions. Delta learning rule can be generalized for the whole network and is the basis of error back-propagation training. The generalized delta rule is explained in section 3.3.3.

3.3 ANN as a Classifier

For patterns that are linearly separable, single layer networks can be used to classify them. But the patterns that are more complex (that cannot be divided into separate classes by decision hyper planes when plotted in the feature space), we need to use more complex networks to classify. Multi-layer feedforward network can be used to learn mapping of any complexity.

3.3.1 Multi layer network

Generalized delta rule can be used for feedforward layered neural networks. For simplicity, three continuous perceptron layers (Figure 2.4) are considered. The network has I input neurons, J hidden neurons and K output neurons. The input pattern is presented to the first layer to produce responses that act as inputs to the second layer. Initially, the weights are assigned randomly. The response of a neuron, which is the fundamental unit of all the layers, can be calculated using the basic definition of the neuron functioning. Let the inputs be $\mathbf{x} = [x1\ x_2\ \ldots\ x_I]$ and the current weights connecting any neuron in the

subsequent layer to the inputs be $\mathbf{w} = [w_1\ w_2 \ldots w_l]^t$. The vector product $\mathbf{w}^t\mathbf{x}$ is the net stimulus for any particular neuron. Being excited by this stimulus, the neuron produces an output which is governed by an activation function like the sigmoid function $f(x)=1/(1+exp(-x))$.

These outputs are produced by all the neurons in the layer being excited and they act as activation for the next layer of neurons (in this case final layer). The process goes on until a final output from the final layer is achieved. Now, supervised training comes into picture. For each input pattern that we present at the input node, there is a predetermined desired response that we want corresponding to that input. This can be a value of 1 for one output node and 0 for all others. The difference in the response that we desired and the one that is actually produced is called the error. It is calculated as $E=1/2*\Sigma (d_k - o_k)^2$, where **d** and **o** are respectively desired and actual response vectors.

This error is used to modify the existing weights in such a manner that the error produced with the modified weights and the current vector *i.e.* the E calculated after modifying is less than that before modifying. This is achieved by negative gradient descent formula. Thus, the network is trained until error becomes less than a threshold or stabilizes to a specific value (*i.e.* rate of decrease with iterations becomes slow).

3.3.2 Momentum method

The objective of the momentum method is to accelerate the convergence of error back-propagation training algorithm. The method employs, in addition to weight adjustment by gradient descent, an extra momentum term that pulls the

network towards faster convergence. A momentum constant (α) is multiplied to the weight correction of the previous step and added to the current step.

$$\Delta w(t) = -\eta \nabla E(t) + \alpha \Delta w(t-1) \tag{3.2}$$

3.3.3 Generalized delta learning rule

Consider the three layer neural network architecture (Figure 2.4) .The first layer is input layer. Second (hidden) layer is input with activation vector **z** and it has corresponding connecting weights v_{ji}. This layer produces output vector **y**, which act as the activation for the subsequent layer, connecting weights being w_{kj}. This layer produces output vector **o**.

The negative gradient descent formula for the hidden layer reads

$$\Delta v_{ji} = -\eta \frac{\partial E}{\partial v_{ji}} \tag{3.3}$$

For $j = 1,2,\ldots J$ and i=1,2,...I

If sum of weight activation product for jth neuron be net_j, then

$$\Delta v_{ji} = \eta \delta_{yj} z_i \tag{3.4}$$

δ_{yj} being the error signal term of the hidden layer having output y. The error signal term is equal to

$$\delta_{yj} = -\frac{\partial E}{\partial y_j} \cdot \frac{\partial y_j}{\partial (net_j)} \tag{3.5}$$

where

$$\frac{\partial E}{\partial y_j} = \frac{\partial}{\partial y_j}\left(\frac{1}{2}\sum_{k=1}^{K}\{d_k - f[(net_k(y)]\}^2\right) \tag{3.6}$$

after calculations we get

$$\frac{\partial E}{\partial y_j} = -\sum_{k=1}^{K}(d_k - o_k)\frac{\partial}{\partial y_j}\{f[net_k(y)]\} \tag{3.7}$$

we can simplify the above expression using the expressions for δ_{ok} and net_k

$$\frac{\partial E}{\partial y_j} = -\sum_{k=1}^{K}\delta_{ok} w_{kj}$$

Putting the values back in equation (3.5) and rearranging

$$\delta_{yj} = f'_j(net_j)\sum_{k=1}^{K}\delta_{ok} w_{kj}, \tag{3.8}$$

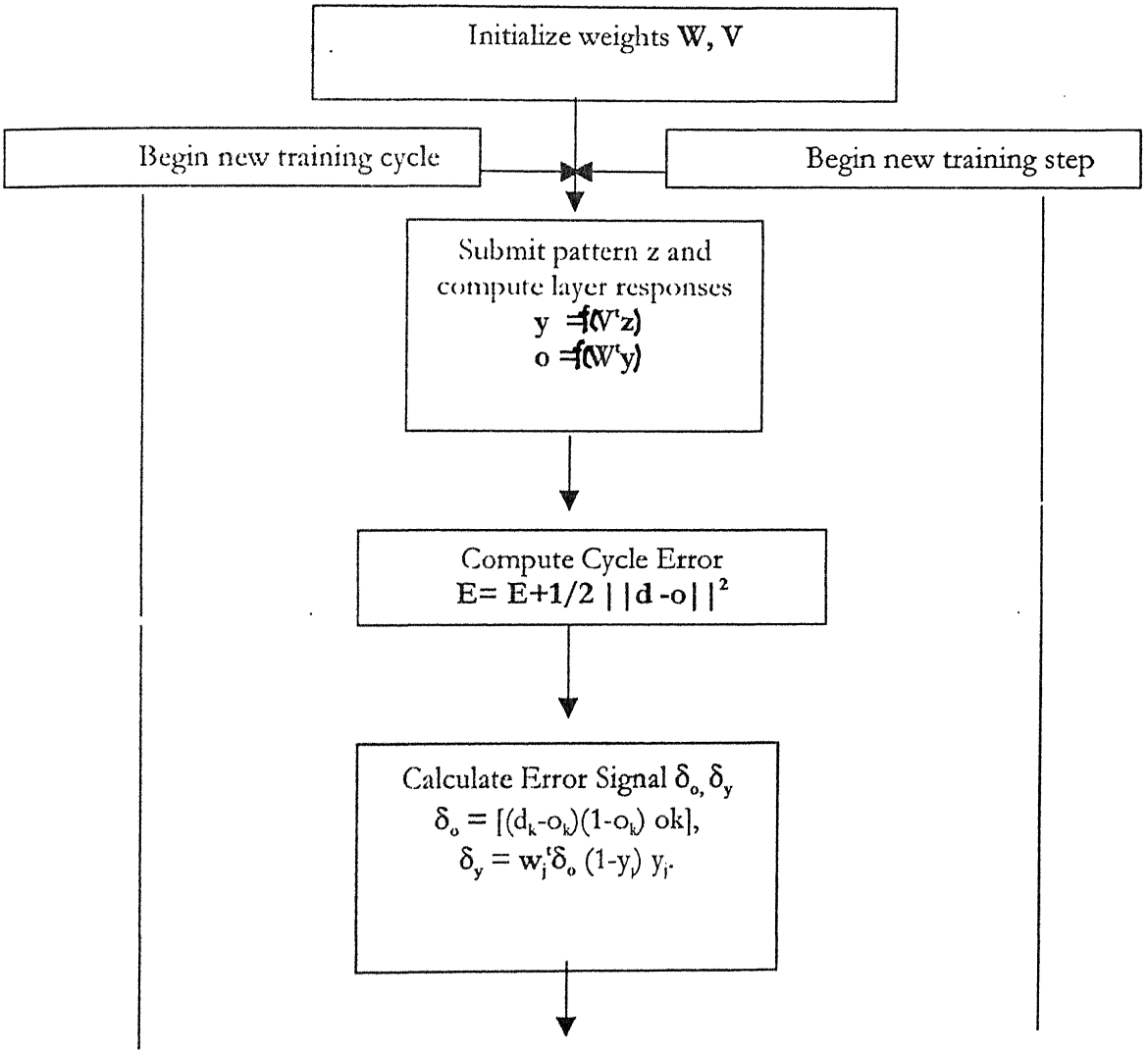
The weight adjustment in the hidden layer now becomes

$$\Delta v_{ji} = \eta f'_j(net_j) z_i \sum_{k=1}^{K}\delta_{ok} w_{kj}, \tag{3.9}$$

This expression expresses the so-called generalized delta-learning rule.

3.3.4 Error back propagation training

The following flow chart depicts the error back propagation training algorithm (adapted from Zurada, 1999)



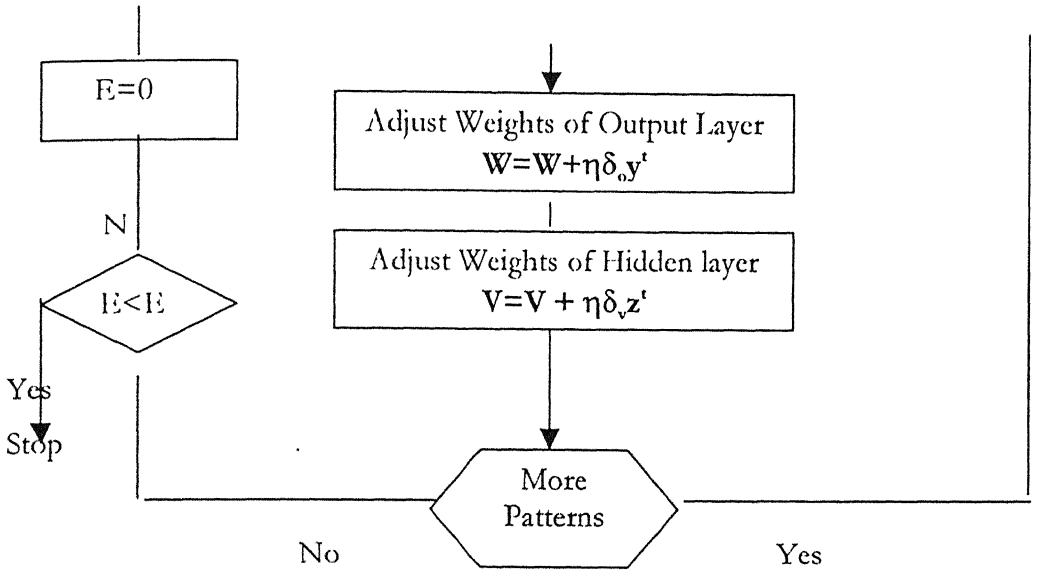


Figure 3.2 Error Back Propagation Training Algorithm (EBPTA) flow chart

4 Methodology

4.0 Introduction

This chapter presents the methodology of the work undertaken. The first section describes sample set selection for various classifications done later. The next section describes various classification experiments performed in the research. The last section describes in detail the use of *JavaNNS* software for classifying remotely sensed images and analysis of results.

4.1 Sample set selection

False color composites of the given areas, generated using NIR, red and green bands were used for the selection of training and testing sample sets. The list of classes for the classification of images for the two study sites is given in Table 4.1 for future reference.

Table 4.1 Classes in the study sites

Class no.	Class abbr.	Class names *(for Lucknow)*	Class no.	Class abbr.	Class names *(for Bhopal)*
1	agr	Agriculture	1	w1	Water1
2	br	Barren land	2	w2	Water2
3	fo	Forest	3	wl	Wetland
4	hr	High residential	4	dv	Dense vegetation
5	hc	High commercial	5	sv	Sparse vegetation
6	mr	Medium residential	6	du	Dense urban
7	prk	Parks	7	mu	Medium urban
8	sc	Scrub	8	sc	Scrub
9	ag2	Agriculture2*	9	br	Barren
10	rv	River			

* Fallow land

Three different sets of training samples and one set of testing samples were selected for each site. The training samples contained of 40, 80 and 120 pixels per class (10 to 30 *n*, *n* being number of bands) as suggested by Mather (1987) and Piper (1992). They are labeled as *train40*, *train80* and *train120*. Testing samples had 79 and 88 pixels per class for the two sites as suggested by equation (2.7). The statistical properties of various sample sets are listed in Appendix A.

4.2 Classifications

Classifications were carried out using Gaussian maximum likelihood classifier as well as a variant of back-propagation artificial neural network (ANN). Three types of classifications were carried out:

1. Classification using Gaussian maximum likelihood classifier using only spectral values as well as using combined spectral and texture (*asm*) feature;
2. Classification using back-propagation ANN with only spectral values: Further, the classification with only spectral values was done in two ways:

   2a. Classification with back-propagation ANN, while using different number of hidden nodes.

   2b. Classification with back-propagation ANN, while using training sets of different sizes.
3. Classification using back-propagation ANN using combined spectral and texture features.

The Gaussian maximum likelihood classification was carried out using *ILWIS* software. The generation of *asm* texture feature was done using program *GLDH.c*. The classification using neural network was carried out using *JavaNNS*. The training method used for the training of the networks was error back-propagation training using momentum term because it ensures faster convergence (Zurada, 1999). Training was stopped when error decrease becomes slow, *i.e.* when error versus iteration curve becomes flat. The learning rate and momentum term for the training of networks were set to 0.2 and 0.5 respectively. The number of iterations for training of various networks was chosen as 1000. Shuffling of patterns alternative was activated, *i.e.* the training patterns were presented to the network in a random fashion.

4.2.1 Classification using Gaussian maximum likelihood classifier

While classifying using Gaussian maximum likelihood classification, no lower threshold on the probabilities was specified so there are no unclassified pixels in the resulting classification. *train80* set was used as the training sample. In GML classification with only spectral values, the four spectral bands (as mentioned in Table 1.1) were used. For GML classification using combined spectral and texture features, a texture band was also used in addition to four spectral bands. This texture band was generated using GLDH from band 1, using *angular second moment* criterion. Asm feature is taken as representative conventional texture approach as the study of Shaban (1999) concludes that results obtained using various different texture features (asm, contrast, entropy etc.) are statistically similar. The window sizes chosen for the generation of texture bands were 3, 5, 7 and 9. The results obtained by this are compared to

corresponding (same window size) window based texture approach discussed in section 4.2.3.

4.2.2 Classification using back-propagation ANN with only spectral values

In classification of images using neural network the normalized spectral values are fed to the input layer of the network, while desired output value depends on the class to which the training vector belongs.

4.2.2.1 Classification with back-propagation ANN, while using different number of hidden nodes.

The training set used for these methods was *train80*. For site 1 (Lucknow), the networks had 4-*n*-10 configurations, where *n* is the number of nodes in the hidden layer. The values of *n* chosen were 6, 8, 10, 12, 14, 16, 18, 20, 22 and 24 for both the cases. For site 2 (Bhopal), networks had 4-*n*-9 configurations. First (input) layer always had four nodes, which is equal to the number of bands while the third (output) layer has ten (or nine, depending upon case study) nodes, equal to the number of classes

4.2.2.2 Classification with back-propagation ANN, while using training sets of different sizes

Three different training sample sets *train40*, *train80* and *train120* were used for the training of networks. A fixed configuration of network for each of the sites was selected and different sample sets were used to train it, one set at a time. The configuration used for case study 1 (*Lucknow*) was 4 -10-10, while for case study 2 (*Bhopal*) configuration was 4 -10-9.

4.2.3 Classification with back-propagation ANN while using combined spectral and texture features

The method of implementation of combined spectral and texture features using window method is explained in Figure 4.1 for a 3 x 3 window case:

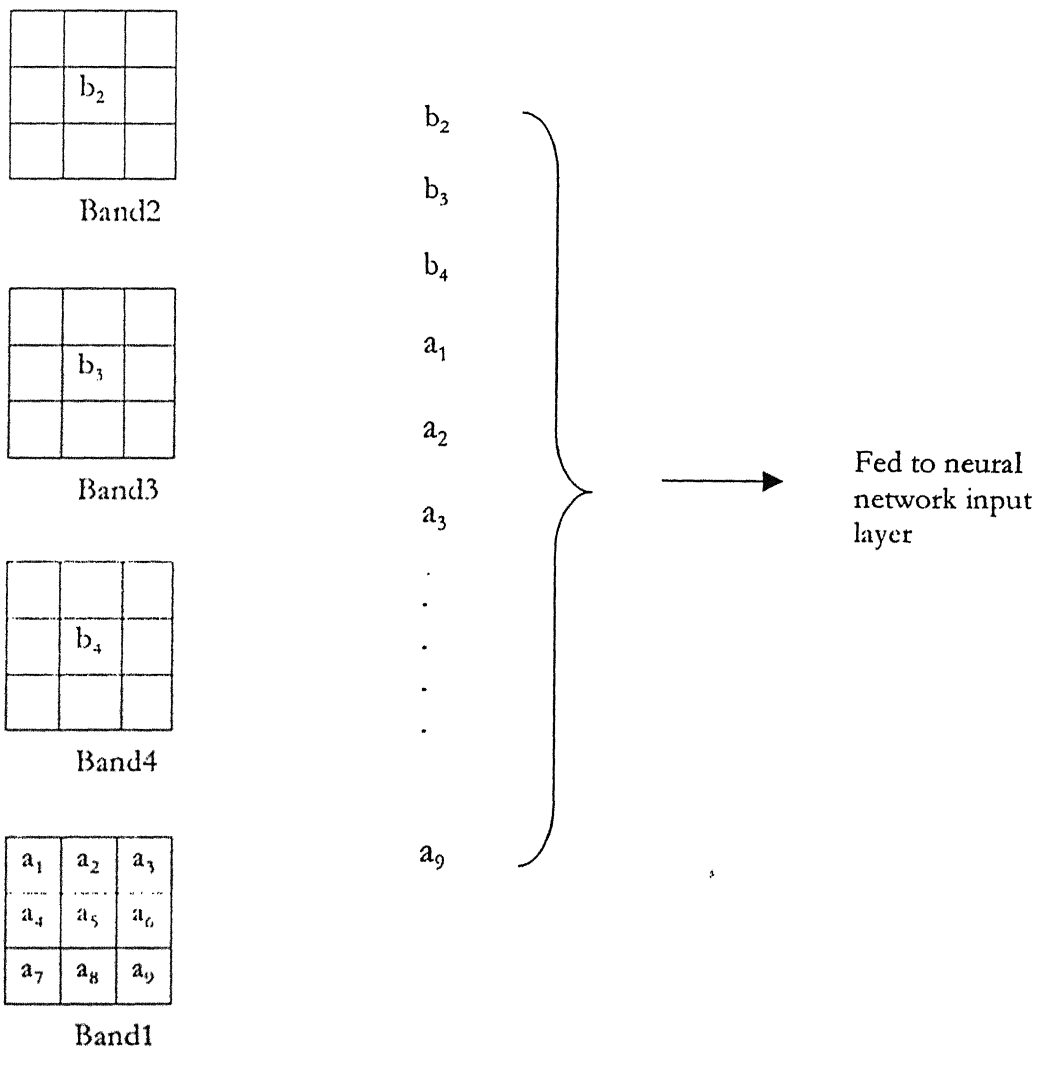


Figure 4.1 Implementation of texture using window method (3 x 3 window example)

Similar procedure is used for higher window sizes. The number of input nodes, when using three pure spectral bands (2, 3 and 4) and one spectral as well as texture band (band 1) is given as:

$$\text{Number of input nodes} = 3 + w^2;$$

where w is the size of the window. For window sizes 3, 5,7 and 9, the number of input nodes are obtained as 12, 28, 52 and 84 respectively.

The number of hidden nodes was kept constant at 10, while the number of output nodes were 10 and 9, for case study 1 (*Lucknow*) and case study 2 (*Bhopal*) respectively. The training sample used for training of the networks was *train80* similar to GML classification.

4.3 Classification of images using *JavaNNS*

The neural network software used to do image analysis is the *JavaNNS* (Originally *SNNS* –Stuttgart Neural Network Simulator: *JavaNNS* is a Java interface written to make the software user friendly). This software can be downloaded from site http://www-ra.informatik.uni-tuebingen.de/SNNS/. *JavaNNS* is a neural network simulator written for general neural network simulation. In order to make it useful for classifying remotely sensed imagery, an interface was developed. This has been done through various codes written in *C* programming language. *JavaNNS* software offers a lot of facilities to define different types of networks and to do operations on them. In this

research, *JavaNNS* is used to implement fully connected feed forward networks and back-propagation with momentum training.

The software takes in a training file (*.pat*) to train a network. This file consists of input activations and corresponding desired output activations. To generate such a file from image files a C program has been written which is called *trainpat.c*. This program takes band images and training image of size 512x512 pixels in *Idrisi .img* {or any other byte (8 bit), binary} format to generate a training file for the network. In the second version (*trainpat2.c*) texture window files (*.txt*) can also be incorporated. The desired output value for a class *i* pixel is a 1 at the *i*th output node and 0 for all other node locations. For example for a pixel belonging to class 3, when the total number of classes is 8, the desired response will be (0 0 1 0 0 0 0 0). Another important file to classify an image is the pattern file for the whole image. It is different from the above-mentioned file in the sense that it does not use any training file i.e. this file does not have any desired outputs. The program that generates this file is *citypat.c* (texture version *citypat2.c*). This program takes band images 512 x 512 pixels in size to generate a full image pattern file. This file can be later used to classify the full image, once the network is trained.

The result file generated from the *JavaNNS* is in *.res* format. It consists of output responses of all the pixels one by one. Output response is the membership values of that pixel in all the classes. A pixel is assigned to a class for which the class membership value is the highest. The program *result.c* takes in *JavaNNS* output *.res* file and/or classified *.img* file and training/or testing *.img* file to generate confusion (error) matrices which can be further used for

statistical analysis using program *stats.c*. A step-by-step procedure for classification using *JavaNNS* is given below.

Steps in classification using *JavaNNS* using spectral features only:

Step 1 Convert images bands into *Idrisi .img* format.

Step 2 Sample set (training and testing) selection using *ILWIS* and its conversion into *Idrisi .img* format

Step 3 Use program *trainpat.c* to generate *JavaNNS* training file.

Step 4 Use program *citypat.c* to generate *JavaNNS* full pattern set file for the whole image.

Step 5 Use *JavaNNS* to define a network and train it using file generated in step3.

Step 6 Load full pattern file (step 4) and save the result in *.res* file.

Step 7 Use program *result.c* to generate error matrix corresponding to the *.res* file generated in step 6 and also for the generation of *JavaNNS* classified image in *Idrisi .img* format

Step 8 Use program *stats.c* to calculate statistics for the error matrix obtained in step 7(a class name (*.txt*) file containing the names of classes should also be supplied).

Steps in classification using *JavaNNS* using combined spectral and texture features:

Step 1(a) Convert images bands into *Idrisi .img* format.

Step 1(b) Use *win.c* to generate window band file (This file has the spectral values for any *nxn* window)

Step 2 Sample set (training and testing) selection using *ILWIS* and its conversion into *Idrisi .img* format.

Step 3 Use program *trainpat2.c* to generate *JavaNNS* training file (Files from step 1 and 2 are used as inputs)

Step 4 Use program *citypat2.c* to generate *JavaNNS* full pattern set file for the whole image.

Step 5 Use *JavaNNS* to define network and train it for file generated in step 3.

Step 6 Load full pattern file (step 4) and save the result in *.res* file.

Step 7 Use program *result.c* to generate error matrix corresponding to the *.res* file generated in step 6 and also for the generation of *JavaNNS* classified image in *Idrisi .img* format.

Step 8 Use program *stats.c* to calculate statistics for the error matrix obtained in step 7 (a class name (*.txt*) file containing the names of classes should also be supplied).

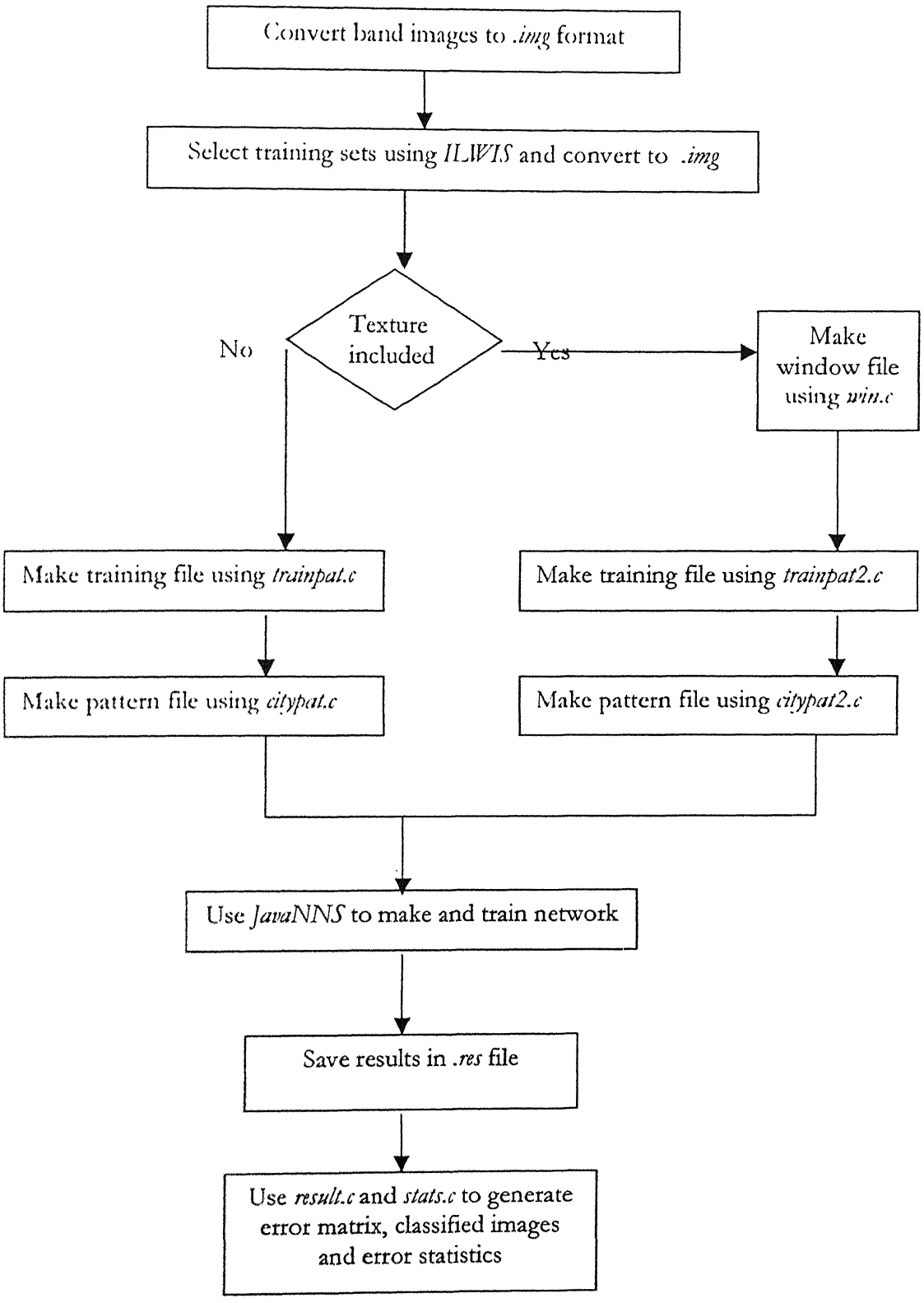


Figure 4.1 Flow chart of classifying images using *JavaNNS*

5 Results and Discussion

5.0 Introduction

Various experiments conducted were explained in previous chapter (section 4.2). The results of those experiments are presented in this chapter. The first section presents results of GML classification. The next section describes the effect of choice of number of nodes on neural network classification accuracy. Generalization property of neural network is discussed in the following section. This is followed by the effect of training set size on classification accuracy. Finally, the effect of combined spectral and texture method on classification accuracy is discussed.

5.1 GML Classification

Results obtained from the GML classification of both the images are presented in Table 5.1.

Table 5.1 Kappa values obtained for GML classification

For *Lucknow*					For *Bhopal*				
Group no.	**Class no.**	**Class abbr.**	**Train**	**Test**	**Group no.**	**Class no.**	**Class abbr.**	**Train**	**Test**
1	1	agr	0.972	0.931	1	1	w1	1.000	1.000
	9	ag2	0.932	1.000		2	w2	1.000	1.000
	10	rv	1.000	1.000		3	wl	1.000	0.928
2	2	br	1.000	0.701		6	du	0.972	0.984
	3	fo	0.919	0.762		8	sc	0.986	1.000
	4	hr	0.835	0.793		9	br	1.000	1.000
	5	hc	0.758	0.662	2	4	dv	0.854	0.762
	6	mr	0.930	0.696		5	sv	0.896	1.000
	7	pk	0.858	0.469		7	mu	0.986	0.461
	8	sc	0.944	0.489					
	Overall		**0.915**	**0.705**		**Overall**		**0.966**	**0.845**

GML classifier has performed well (kappa >0.90), both in case of training and testing, for classes *agriculture*, *agriculture2* and *river* for *Lucknow* site and *water1*, *water2*, *wetland*, *dense urban*, *scrub* and *barren* for *Bhopal* site (Figure 5.1). These classes can be grouped as *group1*. Most of these classes have lesser spectral variation in terms of standard deviation (appendix A). For other classes the accuracies are less (kappa <0.90) for training or testing. Lesser accuracies of these classes can be attributed to their confusion with other classes. For example, in *Lucknow* case class *forest* is confusing with classes *park* and *scrub*. Classes *high residential* and *high commercial* are confusing with each other. Other significant confusions within classes are *park* with *scrub*, and *agriculture2* with *medium residential* and *barren*. In *Bhopal* case, class *sparse vegetation* is getting confused with *dense vegetation*, class *medium urban* is getting confused with *dense urban* and *scrub*. Classes other than *group1* are grouped as *group2*. Most of the *group2* classes show higher values of spectral variation in terms of standard deviation. The GML accuracies of *group2* classes are shown in Figure 5.2.

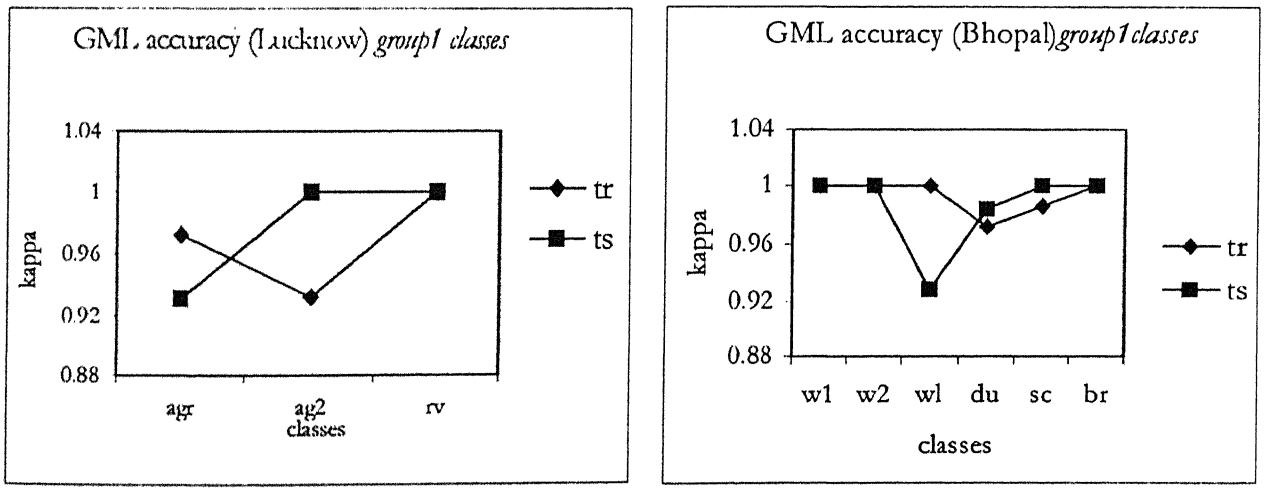


Figure 5.1 GML accuracies of *group 1* classes

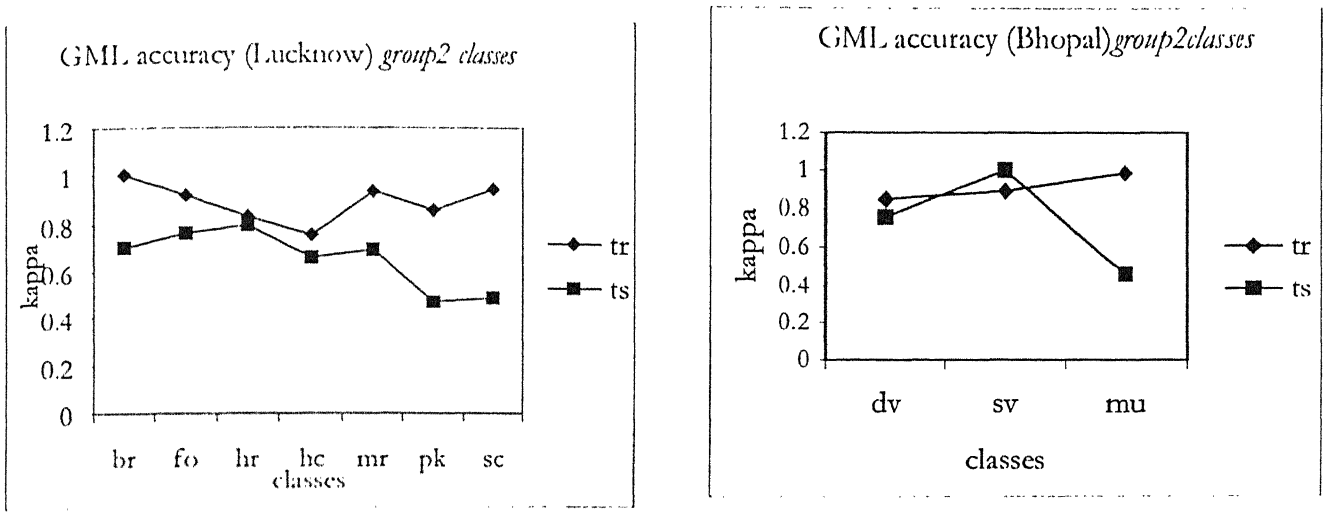


Figure 5.2 GML accuracies of *group2* classes

Overall accuracy of *Bhopal* image is higher as compared to *Lucknow* image. This may be due to the difference in spatial resolution of the two images. Accuracies tend to improve for lower resolution images because of the averaging effect of classes.

5.2 Effect of number of hidden nodes on classification accuracy

The results obtained for varying number of hidden nodes are listed in tables 5.3 and 5.4. The following are the main observations from these results.

(1) For *Lucknow*, *group1* classes show statistically similar† training accuracies to GML classification for all hidden nodes. For testing, however, there is statistically significant drop in accuracy for all hidden nodes except nodes

पुरुषोत्तम काशीनाथ केलकर पुस्तकालय
भारतीय प्रौद्योगिकी संस्थान कानपुर
अवाप्ति क्र० A 141877

† Z value less than 1.96 (for 95% confidence)

12 for class *agriculture*. For classes *agriculture2* and *river*, there is a statistically non-significant drop for all hidden nodes.

(2) Amongst *group2*, classes *barren*, *forest*, *high residential* and *high commercial* show statistically similar accuracy for both training and testing for all hidden nodes. For classes *medium residential* and *park* there is significant improvement in testing accuracy for all hidden nodes. For class *scrub*, hidden nodes 8 and 12 gave statistically poor training results to others, while testing results for all of the hidden nodes were similar to GML classifier.

(3) Amongst *group1* classes for *Bhopal* case, there was a significant drop in testing accuracy of class *wetland* except for hidden nodes 16 and 18. The accuracy for class *dense urban* dropped significantly for training for hidden nodes 8, 16 and 20, while for testing drop was there for all hidden nodes. For other *group1* classes results are statistically similar for all hidden nodes. Thus, it appears that for *group1* classes testing accuracies of ANN classification can be poor as compared to GML classification accuracies.

(4) Amongst *group2* classes for *Bhopal* site, class *sparse vegetation* gave poor testing accuracy for hidden nodes 12 and 18 while all other hidden nodes gave similar accuracy to GML classification. For other classes in *group2* (*dense vegetation* and *medium urban*) accuracies are statistically similar for all hidden nodes, however, testing accuracies are improving (non significantly) for class *medium urban*.

Table 5.2 Kappa and Z-statistics values for *Lucknow* area when using different number of hidden nodes

	Training accuracies											Testing accuracies										
Class abbr	GML a	6* b	8 c	10 d	12 e	14 f	16 g	18 h	20 i	22 j	24 k	GML a	6 b	8 c	10 d	12 e	14 f	16 g	18 h	20 i	22 j	24 k
agr	0.972	0.934	0.934	0.934	0.959	0.946	0.922	0.922	0.922	0.934	0.946	0.931	0.749	0.780	0.808	0.850	0.819	0.775	0.694	0.785	0.783	0.767
ag2	0.932	0.905	0.898	0.922	0.875	0.910	0.910	0.910	0.921	0.910	0.921	1.000	0.917	0.948	0.947	0.928	0.931	0.926	0.924	0.947	0.929	0.972
rv	1.000	0.973	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.934	0.986	0.986	0.986	0.986	0.973	0.973	0.973	0.986	0.986
br	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.701	0.686	0.717	0.706	0.717	0.708	0.708	0.711	0.717	0.708	0.705
fo	0.919	0.843	0.903	0.897	0.878	0.871	0.894	0.882	0.841	0.843	0.874	0.762	0.603	0.662	0.762	0.829	0.749	0.785	0.770	0.560	0.766	0.778
hr	0.835	0.841	0.849	0.843	0.823	0.823	0.841	0.823	0.841	0.821	0.823	0.793	0.763	0.747	0.739	0.719	0.686	0.725	0.722	0.778	0.716	0.719
hc	0.758	0.807	0.807	0.839	0.848	0.841	0.823	0.851	0.828	0.839	0.841	0.662	0.602	0.648	0.691	0.726	0.737	0.707	0.710	0.648	0.700	0.701
mr	0.930	0.886	0.968	0.969	0.939	0.954	0.968	0.968	0.954	0.969	0.941	0.696	0.852	0.968	0.941	0.916	0.983	1.000	0.970	0.941	0.984	0.939
pk	0.858	0.857	0.906	0.867	0.903	0.917	0.863	0.877	0.889	0.900	0.916	0.469	0.837	0.761	0.689	0.716	0.858	0.676	0.801	0.757	0.755	0.786
sc	0.944	0.931	0.843	0.921	0.843	0.886	0.875	0.885	0.906	0.907	0.898	0.489	0.558	0.502	0.512	0.471	0.523	0.510	0.494	0.496	0.533	0.525
overall	**0.915**	**0.899**	**0.910**	**0.919**	**0.906**	**0.914**	**0.910**	**0.911**	**0.910**	**0.911**	**0.915**	**0.705**	**0.738**	**0.748**	**0.752**	**0.757**	**0.772**	**0.752**	**0.750**	**0.748**	**0.764**	**0.762**
		Z Statistics																				
		Z_{ab}	Z_{ac}	Z_{ad}	Z_{ae}	Z_{af}	Z_{ag}	Z_{ah}	Z_{ai}	Z_{aj}	Z_{ak}		Z_{ab}	Z_{ac}	Z_{ad}	Z_{ae}	Z_{af}	Z_{ag}	Z_{ah}	Z_{ai}	Z_{aj}	Z_{ak}
agr		1.114	1.114	1.114	0.443	0.810	1.398	1.398	1.398	1.114	0.810		3.404	2.854	2.318	1.645	2.195	2.905	4.366	2.746	2.772	3.110
ag2		0.605	0.768	0.253	1.218	0.520	0.520	0.520	0.273	0.520	0.273		1.808	1.452	1.453	1.799	1.796	1.800	1.802	1.453	1.797	1.014
rv		1.436	--**	--	--	--	--	--	--	--	--		2.321	1.008	1.008	1.008	1.008	1.436	1.436	1.436	1.008	1.008
br		--	--	--	--	--	--	--	--	--	--		0.225	0.243	0.076	0.243	0.110	0.110	0.146	0.243	0.110	0.065
fo		1.491	0.338	0.471	0.823	0.969	0.518	0.750	1.516	1.491	0.921		1.136	0.622	0.000	0.465	0.087	0.160	0.056	1.406	0.030	0.111
hr		0.098	0.229	0.130	0.201	0.201	0.098	0.201	0.098	0.233	0.201		0.417	0.645	0.752	1.047	1.516	0.953	1.007	0.211	1.089	1.047
hc		0.687	0.711	1.179	1.318	1.219	0.928	1.359	1.012	1.179	1.219		0.778	0.188	0.384	0.845	1.018	0.590	0.628	0.177	0.499	0.523
mr		0.902	1.031	1.064	0.224	0.621	1.031	1.031	0.602	1.048	0.266		2.390	5.062	4.308	3.721	5.543	6.198	5.121	4.308	5.582	4.251
pk		0.007	0.880	0.156	0.820	1.091	0.092	0.327	0.547	0.759	1.061		5.932	4.417	3.348	3.712	6.501	3.154	5.183	4.478	4.409	4.970
sc		0.345	2.116	0.580	2.116	1.304	1.522	1.325	0.886	0.866	1.074		0.906	0.176	0.317	0.252	0.464	0.289	0.068	0.090	0.601	0.489
overall		**1.084**	**0.371**	**0.286**	**0.643**	**0.094**	**0.371**	**0.280**	**0.371**	**0.280**	**0.000**		**1.408**	**1.836**	**2.017**	**2.201**	**2.887**	**2.016**	**1.895**	**1.831**	**2.508**	**2.448**
		Timings (sec)																				
Training		90	109	120	140	156	183	201	227	242	253											
Classification		129	131	135	136	142	146	149	151	149	153											

*number of nodes in the hidden layer

** Z value = 0/0 (both classifications give perfect accuracy)

Table 5.3 Kappa and Z-statistics values for *Bhopal* area when using different number of hidden nodes

	Training accuracies											Testing accuracies										
Class	**GML**	**6***	**8**	**10**	**12**	**14**	**16**	**18**	**20**	**22**	**24**	**GML**	**6**	**8**	**10**	**12**	**14**	**16**	**18**	**20**	**22**	**24**
abbr	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**
w1	1.000	0.986	1.000	0.986	1.000	1.000	0.986	1.000	0.986	1.000	1.000	1.000	0.987	0.987	0.987	1.000	0.987	0.987	0.987	0.987	0.987	0.987
w2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
wl	1.000	1.000	0.986	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.928	0.640	0.634	0.741	0.792	0.816	0.905	0.884	0.684	0.765	0.790
du	0.972	0.909	0.886	0.959	0.934	0.934	0.864	0.934	0.864	0.934	0.934	0.984	0.864	0.816	0.907	0.898	0.844	0.818	0.886	0.811	0.886	0.895
sc	0.986	1.000	1.000	0.972	0.986	0.986	1.000	0.986	0.986	1.000	0.986	1.000	1.000	1.000	0.978	1.000	1.000	1.000	1.000	1.000	1.000	1.000
br	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.985	1.000	0.985	0.985	0.985	0.985	0.985
dv	0.854	0.938	0.925	0.938	0.937	0.938	0.885	0.938	0.938	0.938	0.937	0.762	0.664	0.650	0.688	0.742	0.719	0.743	0.728	0.670	0.729	0.738
sv	0.896	0.847	0.866	0.858	0.836	0.858	0.863	0.858	0.858	0.847	0.836	1.000	0.967	0.982	0.950	0.924	0.965	0.965	0.806	0.967	0.952	0.952
mu	0.986	0.942	0.940	0.957	0.970	0.956	0.953	0.956	0.952	0.957	0.956	0.461	0.441	0.416	0.508	0.565	0.538	0.557	0.564	0.537	0.557	0.526
overall	**0.966**	**0.956**	**0.955**	**0.963**	**0.961**	**0.963**	**0.948**	**0.963**	**0.952**	**0.963**	**0.959**	**0.845**	**0.784**	**0.767**	**0.824**	**0.852**	**0.844**	**0.858**	**0.849**	**0.818**	**0.845**	**0.841**
								Z Statistics														
		Z_{ab}	Z_{ac}	Z_{ad}	Z_{ae}	Z_{af}	Z_{ag}	Z_{ah}	Z_{ai}	Z_{aj}	Z_{ak}		Z_{ab}	Z_{ac}	Z_{ad}	Z_{ae}	Z_{af}	Z_{ag}	Z_{ah}	Z_{ai}	Z_{aj}	Z_{ak}
w1		1.008	--**	1.008	--	--	1.008	--	1.008	--	--		1.007	1.007	1.007	--	1.007	1.007	1.007	1.007	1.007	1.007
w2		--	--	--	--	--	--	--	--	--	--		--	--	--	--	--	--	--	--	--	--
wl		--	1.008	--	--	--	--	--	--	--	--		5.539	5.657	3.640	2.749	2.297	0.541	0.996	4.687	3.218	2.773
du		1.646	2.114	0.416	1.100	1.100	2.533	1.100	2.533	1.100	1.100		2.906	3.790	2.066	2.278	3.273	3.777	2.497	3.933	2.497	2.299
sc		1.008	1.008	0.585	0.009	0.009	1.008	0.009	0.009	1.008	0.009		--	--	1.012	--	--	--	--	--	--	--
br		--	--	--	--	--	--	--	--	--	--		--	--	--	1.008	--	1.008	1.008	1.008	1.008	1.008
dv		1.650	1.369	1.675	1.625	1.675	0.544	1.675	1.675	1.650	1.625		1.300	1.779	1.067	0.292	0.665	0.307	0.504	1.275	0.491	0.356
sv		0.909	0.563	0.715	1.096	0.715	0.616	0.715	0.715	0.909	1.096		1.439	1.010	1.779	2.331	1.441	1.441	1.193	1.439	1.777	1.777
mu		1.389	1.409	1.010	0.605	1.030	1.081	1.030	1.092	1.020	1.030		0.355	0.811	0.811	1.767	1.286	1.588	1.730	1.261	1.621	1.115
overall		**0.864**	**0.998**	**0.300**	**0.445**	**0.300**	**1.514**	**0.300**	**1.261**	**0.300**	**0.587**		**2.944**	**3.707**	**1.071**	**0.370**	**0.073**	**0.672**	**0.221**	**1.347**	**0.000**	**0.219**
								Timings (sec)														
Training		70	90	105	123	143	152	171	189	210	233											
Classification		117	120	122	125	127	132	132	138	140	141											

* number of nodes in hidden layer

** Z value = 0/0 (both classifications give perfect accuracy)

(5) Overall training results for *Lucknow* site are statistically similar to GML classification for all hidden nodes, while for testing, results for hidden nodes 10, 12, 14, 16, 22,and 24 were statistically better than GML classifier. For other nodes testing results were statistically similar to GML classifier. For *Bhopal*, all the nodes are giving statistically similar results to GML classifier both for training and testing, except for nodes 6 and 8, which gave statistically poor results for testing.

The above observations are put in a concise form in Table 5.4. Results not mentioned in this table are statistically similar to GML classification. Some important observations from the section are presented in Figure 5.3 and Figure 5.4.

Table 5.4 Summary of results for ANN classification when using different number of hidden nodes.

Case study	Statistically better *(training)	Statistically poor (training)	Statistically better (testing)	Statistically poor (testing)
Lucknow	*none*	*sc* (nodes 8, 12)	*mr* (all nodes)	*agr*, all except 12 nodes
			pk (all nodes)	*rv* (nodes 6)
			Overall (all except nodes 6, 8, 18, 20)	
Bhopal	*none*	*du* (nodes 8, 16 and 20)	*none*	*wl* (all nodes except 16, 18)
				sv (nodes 12)
				Overall (nodes 6, 8)

*as compared to Gaussian maximum likelihood classifier

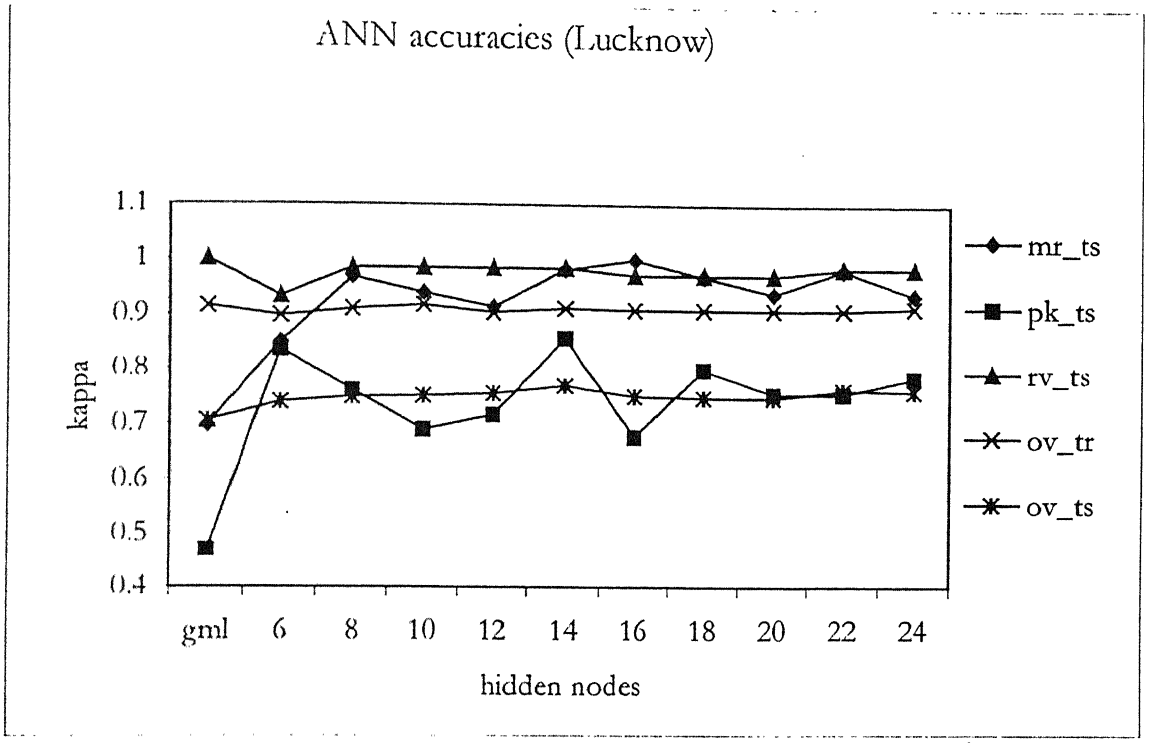


Figure 5.3 ANN accuracies for *Lucknow* site (selected classes)

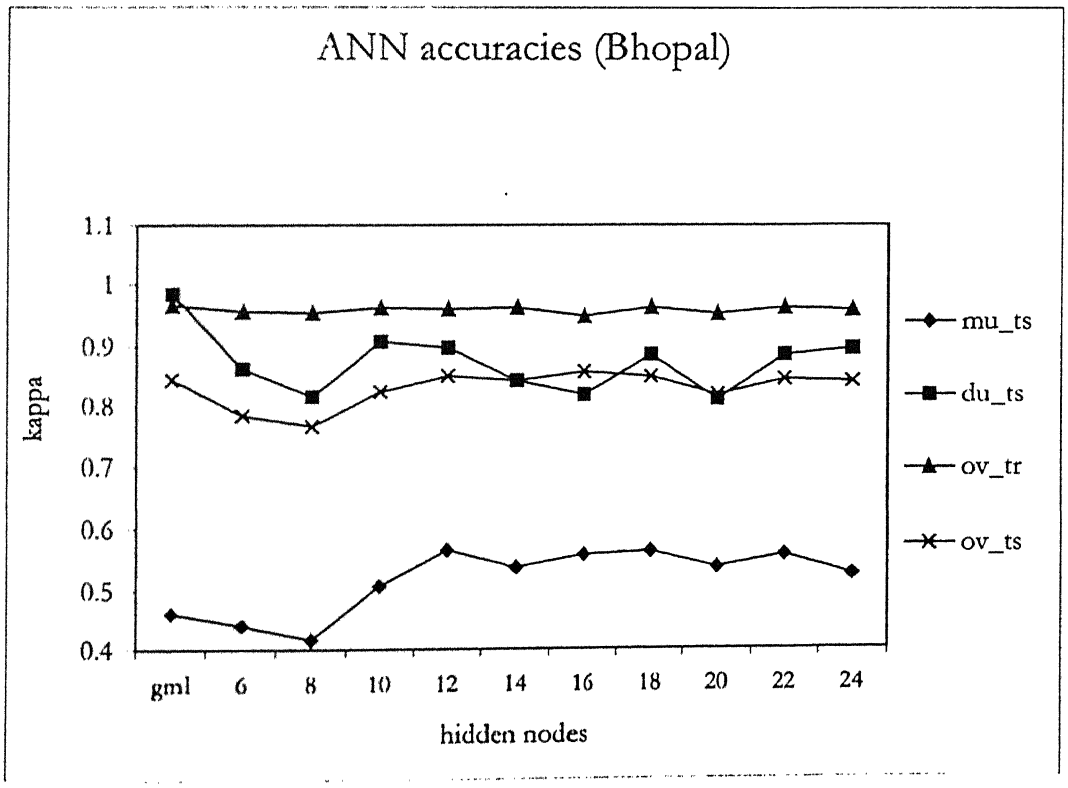


Figure 5.4 ANN accuracies for *Bhopal* site (selected classes)

Thus, for *group1* classes neural network methods may give lesser accuracy as compared to GML classification, while for *group2* ANN methods are favorable. Among neural networks with different number of hidden nodes, performance of hidden nodes 6 and 8 is poor, however, other hidden nodes are giving similar performance; so the choice of 10 hidden nodes given by Equation (2.6) as optimal is correct. The results are in agreement to the observations by Foody and Arora (1997), that network architecture does not have a significant effect on accuracy

5.3 Generalization property of the neural network

Various training and testing accuracy values for two classification schemes (GML and ANN) are listed in Table 5.5. Network with hidden nodes 10 is taken for comparison because in the previous analysis it has been shown to be optimal. It can be observed from the table that neural networks have better generalization capability as compared to GML for classes *medium residential* and park for *Lucknow* and *medium urban* for *Bhopal.* Thus, we see that for *group1* classes neural network does not have a better generalization ability for most of the cases, while amongst *group2* classes, better generalization is obtained for some classes while for some it is worse.

Table 5.5 Comparison of GMLC and ANN for generalization ability.

For *Lucknow* area							For *Bhopal* area						
	GML			ANN (10 nodes)				GML			ANN (10 nodes)		
Class abbr.	Train κ (a) %	Test κ (b) %	a-b %	Train κ (a) %	Test κ (b) %	a-b %	Class abbr.	Train κ (a) %	Test κ (b) %	a-b %	Train κ (a) %	Test κ (b) %	a-b %
agr	97.2	93.1	4	93.4	80.8	12	w1	100.0	100.0	0	98.6	98.7	0
ag2	93.2	100.0	7	92.2	94.7	3	w2	100.0	100.0	0	100.0	100.0	0
rv	100.0	100.0	0	100.0	98.6	1	wl	100.0	92.8	7	100.0	74.1	26
br	100.0	70.1	30	100.0	70.6	29	du	97.2	98.4	1	95.9	90.7	5
fo	91.9	76.2	16	89.7	76.2	14	sc	98.6	100.0	1	97.2	97.8	0
hr	83.5	79.3	5	84.3	73.9	10	br	100.0	100.0	0	100.0	100.0	0
hc	75.8	66.2	10	83.9	69.1	14	dv	85.4	76.2	9	93.8	68.8	25
mr	93.0	69.6	23	96.9	94.1	3	sv	89.6	100.0	10	85.8	95.0	9
pk	85.8	46.9	39	86.7	68.9	18	mu	98.6	46.1	53	95.7	50.8	45
sc	94.4	48.9	45	92.1	51.2	41							
overall	**91.5**	**70.5**	**21**	**91.9**	**75.2**	**17**	overall	**96.6**	**84.5**	**12**	**96.3**	**82.4**	**14**

5.4 Effect of training set size on accuracy

Testing performances of various training schemes are listed in table 5.6. The following conclusions can be derived from the table.

(1) Sample sets *train40* and *train80* gave similar overall results for both of the cases. For *train120* results are poor as compared to earlier two, however results are significant for *Bhopal* only(Figure 5.5).

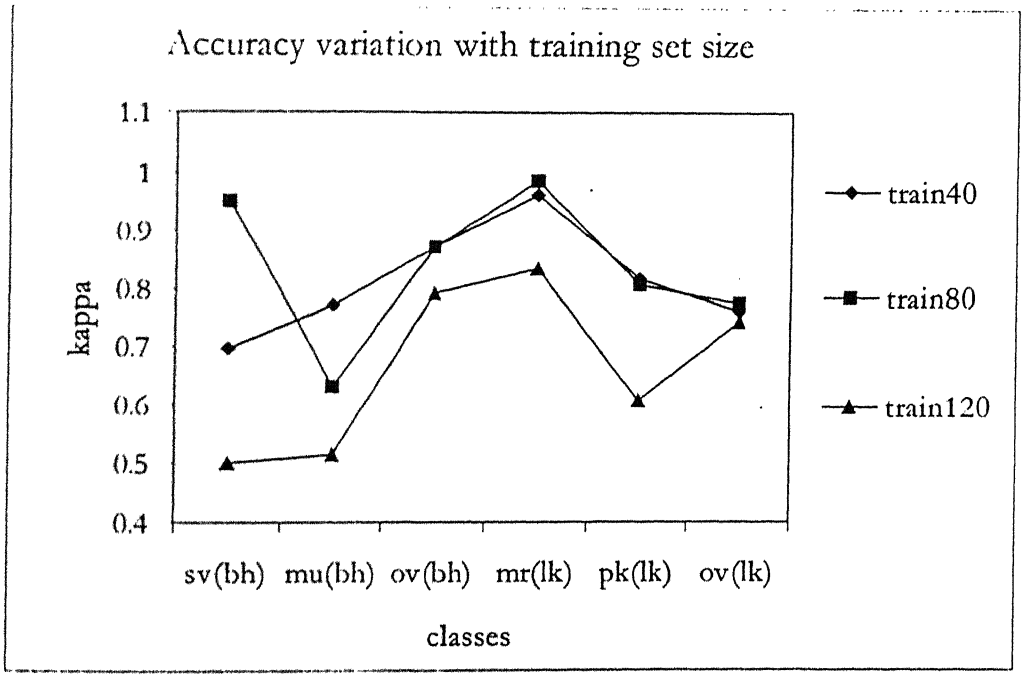
(2) From a class wise analysis of *Lucknow* case, *medium residential* and *park* classes are giving statistically poor results for *train120*, for *river* class (no. 10), sample set *train40* is giving statistically poor results to *train120*. For other classes difference is statistically non significant. For *Bhopal* case, classes *wetland* and *dense urban* gave better results for *train120*. For *medium urban* class *train40* gave better results while for *barren* class *train120* gave

poor results statistically to other sets. For other classes results are statistically similar.

Table 5.6 Testing set kappa values when using different sized training sets.

Class abbr.	40* (a)	80 (b)	120 (c)	Z_{ab}	Z_{ac}	Z_{bc}	Class abbr.	40 (a)	80 (b)	120 (c)	Z_{ab}	Z_{ac}	Z_{bc}
agr	0.872	0.803	0.775	1.221	1.657	0.451	w1	0.987	0.987	0.987	0.000	0.000	0.000
ag2	0.847	0.907	0.948	0.912	1.660	0.722	w2	1.000	1.000	1.000	--	--	--
rv	0.932	0.959	1.000	0.730	2.323	1.772	wl	0.874	0.835	0.963	0.748	2.154	2.902
br	0.711	0.714	0.659	0.038	0.737	0.768	du	0.855	0.844	0.979	0.189	2.848	3.033
fo	0.861	0.710	0.847	1.575	0.150	1.365	sc	1.000	1.000	1.000	--	--	--
hr	0.607	0.732	0.698	1.826	1.284	0.461	br	0.986	0.985	0.872	0.063	2.981	2.912
hc	0.663	0.734	0.684	0.824	0.257	0.658	dv	0.700	0.755	0.616	0.758	0.864	1.514
mr	0.961	0.984	0.833	0.739	2.488	3.250	sv	0.698	0.952	0.502	4.495	2.907	8.599
pk	0.815	0.806	0.607	0.144	3.210	3.003	mu	0.770	0.632	0.516	2.137	4.077	1.802
sc	0.596	0.548	0.626	0.726	0.432	1.105							
overall	**0.761**	**0.775**	**0.743**	**0.623**	**0.792**	**1.413**		**0.872**	**0.872**	**0.790**	**0.000**	**4.119**	**4.119**

*size of training sample set



*bh=*Bhopal*, lk=*Lucknow*, ov= Overall.

Figure 5.5 ANN accuracies for different training set sizes (selected classes)

Thus, increasing the training set size is not improving accuracy. This is in agreement with Heerman and Khazenie's (1992) finding that accuracy did not improve significantly for larger training sets.

5.5 Effect of using texture

All texture methods (described in section 4.2.3) significantly improved accuracy for many classes. The kappa values and Z statistics obtained for various texture methods is listed in Table 5.7 and 5.8. The following are the salient observations from the results:

(1) For *Lucknow* site, overall training and testing results obtained using window sizes 5, 7 and 9 are statistically better than those obtained using corresponding (same window size) *asm* methods, while window size 3 gives statistically better results for testing set and similar results for training set. For *Bhopal* site windows 7 and 9 are giving better results for training as compared to corresponding *asm* methods, however, testing performance of window 9 is poor. Other results are statistically similar.

(2) Amongst *group2* classes texture window methods are found to be statistically significantly improving accuracies over corresponding *asm* method for classes *high commercial*, *medium residential* and *park* for *Lucknow* case and class *medium urban* for *Bhopal* case (Figure 5.6 and Figure 5.7). The best window size for *high commercial*, *medium residential* and *park* is 7, 3 and 5 respectively, for *Lucknow* case and 5 for *medium urban* in *Bhopal* case. In *Bhopal* case, window texture methods are significantly reducing testing accuracy of class *sparse vegetation*. This may be because *sparse vegetation* is present surrounding

Table 5.7 Comparison of conventional (*asm*) and window texture methods for *Lucknow* case study.

Class abbr.	Training accuracies								Testing accuracies							
	Asm3*	W3**	Asm5	W5	Asm7	W7	Asm9	W9	Asm3	W3	Asm5	W5	Asm7	W7	Asm9	W9
agr	0.972	0.986	0.972	1.000	0.972	1.000	0.986	1.000	0.884	0.986	0.874	0.922	0.766	0.946	0.649	0.850
ag2	0.945	0.922	0.959	1.000	0.959	1.000	0.959	1.000	1.000	0.828	1.000	0.893	1.000	0.759	1.000	0.940
rv	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.973	1.000	0.959	1.000	0.959	1.000	0.973	1.000	0.893
br	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.687	0.722	0.717	0.717	0.788	0.710	0.807	0.711
fo	0.959	0.890	0.986	0.972	1.000	1.000	1.000	0.986	0.899	0.810	1.000	0.722	1.000	0.812	0.630	0.533
hr	0.828	0.805	0.873	0.960	0.931	1.000	0.944	0.986	0.772	0.646	0.873	0.823	0.984	0.717	0.948	0.690
hc	0.766	0.915	0.817	1.000	0.902	1.000	0.931	1.000	0.618	0.910	0.698	0.924	0.696	0.943	0.643	0.818
mr	0.931	0.970	0.944	0.986	0.944	1.000	0.944	0.911	0.716	0.904	0.673	0.885	0.596	0.831	0.460	0.750
pk	0.887	0.923	0.920	1.000	0.932	1.000	0.973	0.866	0.472	0.719	0.487	0.894	0.444	0.696	0.426	0.745
sc	0.986	0.843	0.986	0.946	0.972	1.000	0.972	1.000	0.457	0.489	0.486	0.540	0.425	0.716	0.433	0.717
overall	**0.928**	**0.922**	**0.946**	**0.986**	**0.961**	**1.000**	**0.971**	**0.969**	**0.692**	**0.768**	**0.713**	**0.812**	**0.693**	**0.806**	**0.626**	**0.745**

	Z statistics															
	Z33†		Z55		Z77		Z99		Z33		Z55		Z77		Z99	
agr	0.575		1.436		1.436		1.008		2.623		0.994		3.458		3.311	
ag2	0.562		1.772		1.772		1.772		3.303		2.368		4.145		1.458	
rv	--		--		--		1.436		1.772		1.772		1.436		3.008	
br	--		--		--		--		0.534		0.003		1.161		1.437	
fo	1.587		0.585		--		1.008		0.751		4.236		3.693		0.317	
hr	0.383		1.907		2.325		1.386		1.795		0.867		5.456		4.048	
hc	2.371		4.013		.2.797		2.325		4.561		3.792		4.586		4.783	
mr	1.081		1.386		2.063		0.808		2.936		3.231		3.129		3.404	
pk	0.709		2.563		2.323		2.544		3.805		6.974		3.783		1.458	
sc	3.400		1.329		1.436		1.436		0.420		0.748		3.820		3.008	
overall	**0.399**		**4.231**		**5.387**		**0.155**		**3.215**		**4.372**		**4.911**		**4.866**	

Timings (sec)								
Training	148		212		314		420	
Classification	142		152		166		188	

* spectral (b2, b3, b4) + asm3x3 (b1)

**spectral (b2, b3, b4) + win 3x3 (b1)

† Zii refers to Z statistics between asm (i) and win (i).

Table 5.8 Comparison of conventional (*asm*) and window texture methods for *Bhopal* case study.

Class abbr.	Training accuracies								Testing accuracies							
	Asm3*	W3**	Asm5	W5	Asm7	W7	Asm9	W9	Asm3	W3	Asm5	W5	Asm7	W7	Asm9	W9
w1	1.000	1.000	1.000	0.986	1.000	1.000	1.000	0.986	0.987	1.000	1.000	0.987	1.000	1.000	1.000	0.987
w2	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.940	1.000	0.975	1.000
wl	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.927	0.917	0.928	0.916	0.885	0.975	0.846	1.000
du	0.972	0.874	0.972	0.863	0.958	0.934	0.972	0.972	0.983	0.867	0.940	0.858	0.985	0.913	0.911	0.934
sc	0.986	0.946	1.000	0.973	1.000	0.973	1.000	1.000	1.000	0.914	1.000	0.901	1.000	0.930	0.983	1.000
br	1.000	1.000	0.986	1.000	1.000	1.000	1.000	1.000	1.000	0.984	0.960	0.987	0.960	0.987	0.891	1.000
dv	0.869	0.893	0.879	1.000	0.869	1.000	0.890	1.000	0.771	0.674	0.807	0.876	0.737	0.831	0.689	0.841
sv	0.925	0.913	0.912	0.875	0.939	1.000	0.900	1.000	1.000	0.873	1.000	0.502	1.000	0.559	1.000	0.374
mu	0.986	0.952	1.000	1.000	0.972	1.000	0.986	0.945	0.465	0.757	0.536	0.953	0.564	0.943	0.711	0.646
overall	**0.970**	**0.952**	**0.972**	**0.963**	**0.970**	**0.989**	**0.972**	**0.989**	**0.847**	**0.879**	**0.874**	**0.859**	**0.859**	**0.888**	**0.871**	**0.791**
Z statistics																
		Z33†		**Z55**		**Z77**		**Z99**		**Z33**		**Z55**		**Z77**		**Z99**
w1		--		1.008		--		1.008		1.007		1.007		--		1.007
w2		--		--		--		--		--		--		2.314		1.434
wl		--		--		--		--		0.250		0.290		2.354		4.105
du		2.343		2.562		0.664		0.013		2.845		1.725		2.051		0.506
sc		1.349		1.436		1.436		--		2.779		1.821		1.796		1.009
br		--		1.008		--		--		1.008		1.062		1.045		3.195
dv		0.453		3.218		3.414		3.013		1.528		1.257		1.584		2.548
sv		0.248		0.746		2.069		2.800		2.836		11.500		9.377		16.521
mu		1.102		--		1.437		1.349		4.725		8.320		7.462		0.987
overall		**1.730**		**0.940**		**2.380**		**2.223**		**1.777**		**0.781**		**1.598**		**3.991**
Timings (sec)																
Training		140		185		288		423								
Classification		125		138		147		200								

* spectral (b2, b3, b4) + asm3x3 (b1)

**spectral (b2, b3, b4) + win 3x3 (b1)

† Zii refers to Z statistics between asm (i) and win (i).

many other classes like *scrub* and *medium urban* and due to strong neighborhood interpretation of window-based approach, these classes are getting classified as *sparse vegetation.*

(3) For *group1* classes, similar accuracies as compared to corresponding *asm* methods are obtained for most of the cases, however, *agriculture2* class in *Lucknow* image is giving statistically poor testing accuracies, while class *agriculture* is giving better testing accuracies for window based methods as compared to corresponding *asm* methods.

Thus, window based texture approach is found useful for classes like *high commercial, medium residential, park and medium urban* while for classes that are more uniform like *river, water* etc. conventional texture approaches are favorable. Thus, it appears that window based texture approach captures neighborhood information in a better manner as compared to conventional texture methods. Overall texture results using window texture methods are better as compared to corresponding *asm* methods in *Lucknow* case, however for *Bhopal* case the results are similar for most of the cases. This may be because of the fact that *Bhopal* is dominated by *group1* classes. Window sizes 5 and 7 are found better as compared to other windows.

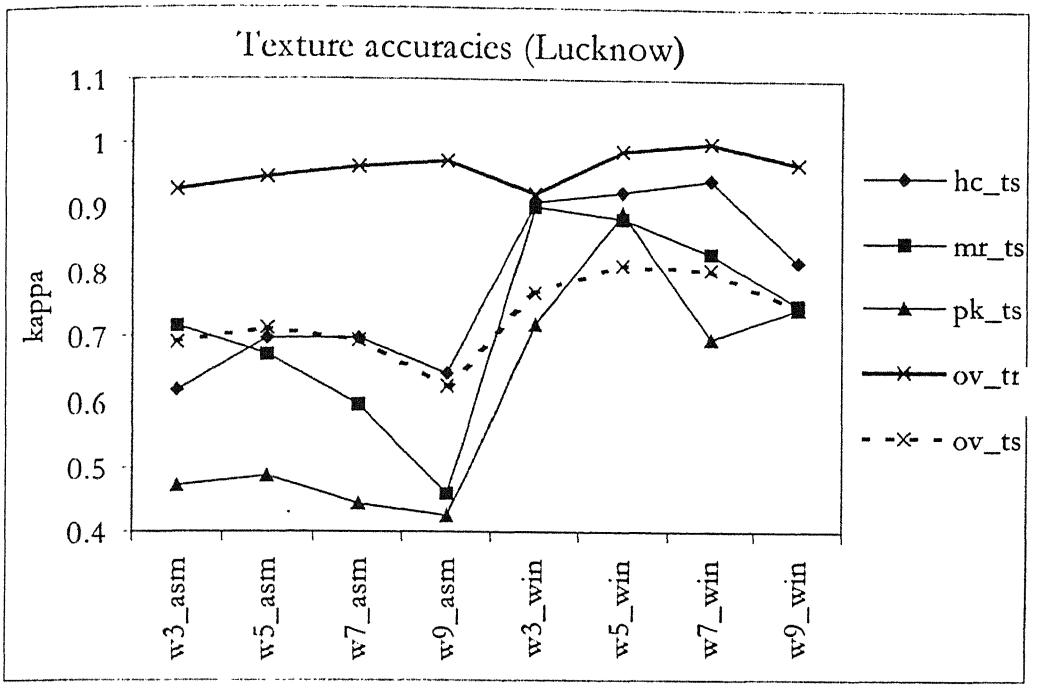


Figure 5.6 Texture accuracies for *Lucknow* site using conventional and window approaches (selected classes)

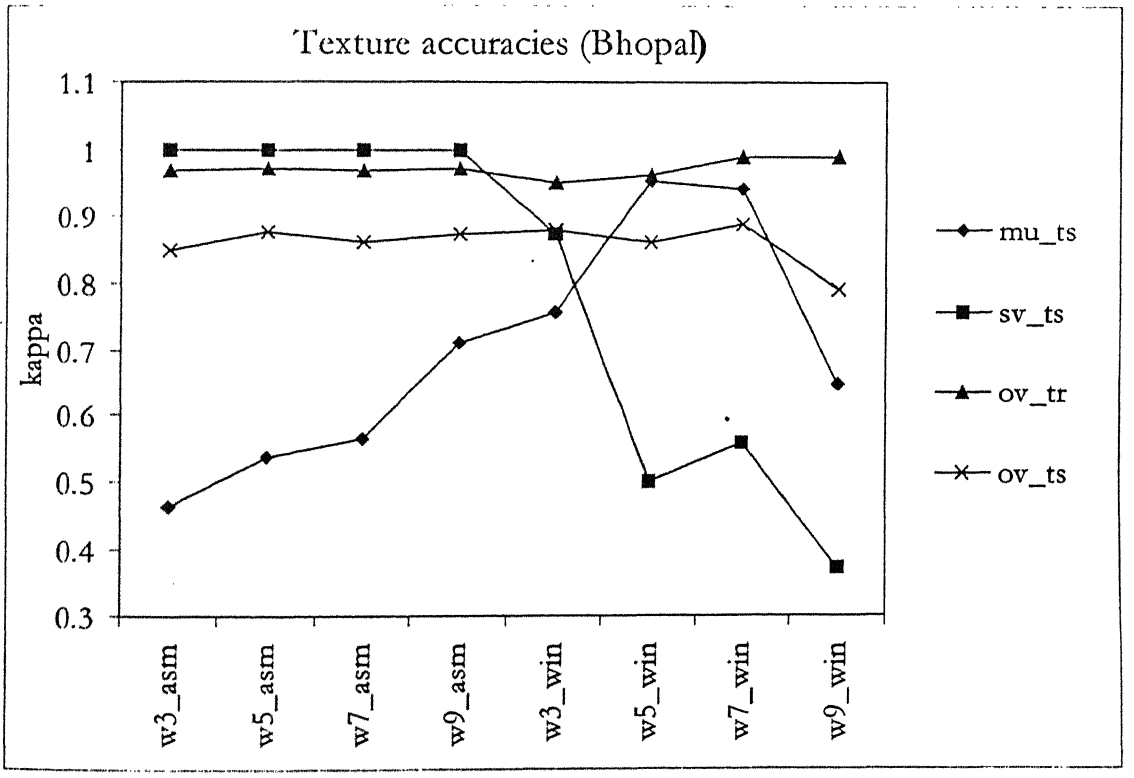


Figure 5.6 Texture accuracies for *Bhopal* site using conventional and window approaches (selected classes)

6 Conclusion

6.1 Conclusions

The following conclusions can be made from the research:

(1) The number of nodes in the hidden layer does not have much effect on classification. However nodes 6 and 8 give poor testing results as compared to other nodes. Hence, a minimum of 10 nodes should be selected. This is in agreement with the number of optimum nodes derived from theoretical considerations.

(2) ANN gives better classification results as compared to GML classifier for some classes that are non-uniform in nature while for uniform classes GML is found to be good enough and ANN may give similar or lesser accuracy as compared to GML classifier.

(3) Accuracy does not improve significantly with increase in training set size; in fact it can even go down. Thus, minimal training set ($10n$) is good enough for classification.

(4) Window based texture approach is able to classify more accurately spectrally non-uniform classes, while for uniform classes conventional texture approach is found better. It may be because of better interpretation of neighborhood information by window-based approach.

(5) Overall, window sizes 5 and 7 are found to be better for classification of urban environments as compared to other windows.

6.2 Recommendations for future work

Output of *JNNS* is fuzzy in nature, i.e. membership values of various classes for a particular pixel are given. While doing hard classification over these values, maximum of these membership values is selected; thus information about other membership values is lost. Neuro-fuzzy classification techniques can be used, so that the fuzzy information available as the output of the neural network is utilized.

References

Amari, S. I., 1990, Mathematical foundations of neurocomputing, *IEEE Proc,* 78(9), 1443-1463.

Atkinson, P. M., and Tatnall, A. R. L., 1997, Neural networks in remote sensing, *International Journal of Remote Sensing,* **18**, 699-709.

Benediktsson, J.A., Swain and P.H., Erosy, 1990, Neural network approaches versus statistical methods in classification of multi source remotely sensed data, *IEEE Transaction on Geoscience and Remote Sensing,* **28,** 540-551.

Bischof, H., Schneider, W. and Pinz, A.J., 1992, Multispectral classification of Landsat-images using neural networks, *IEEE Transaction on Geoscience and Remote Sensing,* **30,** 482-490.

Blonda, P., La Forgia, V., Pasquariello, G., and Satalino, G., 1994, Multispectral classification by a modular neural network architecture, *Proceedings of IGARSS'94, Pasadena, CA,* (Piscataway, NJ:I.E.E.E.), 1873-1876.

Civco, D. L., 1991, Landsat TM image classification with an artificial neural network, *in Proc. ASPRS-ACSM Annual meeting,* Baltimore, MD, **3**, 67-77.

Congalton R.G., and Green K., 1999, *Assessing the Accuracy of Remotely Sensed Data: Principles and Practices*, Lewis Publishers: FL.

Dreyer, P., 1993, Classification of land cover using optimized neural nets on SPOT data, *Photogrammetric Engineering and Remote Sensing*, **59**(5), 617-621.

Dikshit, O., 1992, The Classification of texture in remotely sensed environmental imagery, *Ph.D. Thesis*, University of Cambridge, U.K.

Fierens, F., Kanelloupoulos, I., Wilkinson, G.G. and Megier, J., 1994, Comparison and visualization of feature space behavior of statistical and neural classifiers of satellite imagery, *Proceedings of IGARSS'94, Pasadena, CA,* (Piscataway, NJ:I.E.E.E.),1880-1882.

Foody, G. M., and Arora, M.K., 1997, An evaluation of some factors affecting the accuracy of classification by an artificial neural network, *International Journal of Remote Sensing*, **18**, 799-810.

Foody, G. M. et al, 1995, The effect of training set size and composition on artificial neural network classification, *International Journal of Remote Sensing*, **16**, 1707-1723.

Foody, G. M., 1996, Relating the land cover composition of mixed pixels to artificial neural network classification output, *Photogrammetric Engineering and Remote Sensing*, **62**(5), 491-499.

Gool, L. V., Dewaele, P., and Qosterlinck, A., 1985, Texture analysis, anno 1983. *Computer Vision, Graphics, and Image Processing,* **29**, 336-357.

Haralick, R.M., 1979, Statistical and Structural Approaches to Texture, *Proceedings of the IEEE*, Vol.**67**, No.5, 786-803.

Hara, Yoshihisa, Robert G. Atkins, Simen H. Yuch, Robert T. Shin and J.A.Kang, 1994, Application of neural networks to radar image classification, *IEEE Transactions on Geoscience and Remote Sensing,* **32**(1), 100-109.

Heermann, P. D., and Khazenie, N., 1992, Classification of multi spectral remote sensing data using a back-propagation neural network, *IEEE Transaction on Geoscience and Remote Sensing,* **30**, 81-88.

Hepner, G. F., Logan T., Ritter N., and Bryant, N., 1990, Artificial neural network classification using a minimal training set: comparison to conventional supervised classification, *Photogrammetric Engineering and Remote Sensing,* **56**, 469-473.

IRS 1C Data Users Handbook, NRSA, Hyderabad.

Jain, A.K., 1989, *Fundamentals of Digital Image Processing, Prentice* Hall, Englewood Cliffs, N.J., U.S.A.

Jensen, J. R., 1996, *Introductory Digital Image Processing: A Remote Sensing Perspective,* Prentice Hall, Englewood Cliffs, NJ, U.S.A.

Kanellopoulos, I., and Wilkinson, G. G., 1997, Strategies and best practice for neural network image classification, *International Journal of Remote Sensing*, **18**, 711-725.

Kanellopoulos, I. *et al*, 1992, Land cover discrimination in SPOT HRV imagery using an artificial neural network –a 20 class experiment, *International Journal of Remote Sensing*, **13**, 917-924.

Kiang, R. K., 1992, Classification of remotely sensed data using OCR-inspired neural network techniques, *Proceedings of IGARSS'92, Houston, TX*, (Piscataway, NJ:I.E.E.E.), 1081-1083.

Liu, Z. K. and Xiao, J.Y., 1991, Classification of remotely sensed image data using artificial neural networks, *International Journal of Remote Sensing*, **12**,2433-2438.

Mather, P.M., 1987, *Computer Processing of Remotely Sensed Images*, (Chichester: Wiley)

Pao, Y.H., 1989, *Adaptive Pattern Recognition and Neural Networks* (Reading, MA: Addison-Wesley).

Paola, J. D., and Schowengerdt, R. A., 1995, A detailed comparison of back-propagation neural network and maximum-likelihood classifier for urban land use classification, *IEEE Transactions on Geoscience and Remote Sensing*, **33**, 981-995.

Appendix A

Table A .1 Sample set characteristics (*Lucknow*): train80

Class	avb1	avb2	avb3	avb4	sd1	sd2	sd3	sd4	avsd*	cv1	cv2	cv3	cv4	avcv*
agr	98.71	67.04	89.10	156.21	5.00	6.24	5.59	11.18	5.61	5.06	9.30	6.28	7.16	6.88
ag2	107.49	78.65	69.61	145.16	2.99	2.54	2.67	4.07	2.73	2.78	3.22	3.83	2.81	3.28
rv	88.28	50.64	26.91	77.76	3.00	1.75	2.11	18.95	2.29	3.40	3.46	7.83	24.37	4.90
br	116.81	93.11	81.76	179.07	6.51	5.91	5.49	7.27	5.97	5.57	6.35	6.71	4.06	6.21
fo	70.64	35.16	87.03	98.63	1.71	1.40	2.37	2.05	1.83	2.42	3.98	2.72	2.08	3.04
hr	99.94	68.53	56.14	109.57	4.58	4.21	3.36	3.32	4.05	4.59	6.15	5.98	3.03	5.57
hc	90.91	59.70	60.91	105.63	9.18	8.30	6.25	5.97	7.91	10.10	13.91	10.26	5.66	11.42
mr	111.32	79.16	72.26	132.04	6.77	5.90	4.53	6.47	5.73	6.08	7.46	6.27	4.90	6.60
pk	74.26	41.45	75.91	97.49	5.05	5.16	7.37	7.92	5.86	6.80	12.46	9.71	8.12	9.66
sc	69.90	35.78	93.13	84.16	1.56	0.94	4.65	11.43	2.38	2.24	2.63	4.99	13.58	3.29

Table A .2 Sample set characteristics (*Lucknow*): test79

Class	avb1	avb2	avb3	avb4	sd1	sd2	sd3	sd4	avsd*	cv1	cv2	cv3	cv4	avcv*
agr	94.95	63.32	88.99	162.08	7.16	8.48	4.35	18.54	6.66	7.54	13.40	4.88	11.44	8.61
ag2	110.25	80.27	72.44	154.41	6.14	7.59	5.03	16.76	6.25	5.57	9.46	6.95	10.86	7.33
rv	86.09	50.25	27.21	69.78	5.91	2.65	1.83	21.32	3.46	6.86	5.28	6.71	30.55	6.28
br	119.71	93.49	85.41	174.70	5.69	5.51	5.65	7.78	5.62	4.75	5.90	6.62	4.46	5.76
fo	69.70	36.29	90.30	95.33	2.53	2.06	6.91	3.66	3.83	3.63	5.69	7.65	3.84	5.66
hr	97.08	66.11	55.34	107.43	4.40	3.72	3.71	3.80	3.94	4.53	5.62	6.70	3.54	5.62
hc	91.51	60.49	58.52	101.68	9.45	7.12	7.09	7.10	7.89	10.33	11.76	12.12	6.98	11.40
mr	106.11	73.76	70.52	125.72	6.39	5.48	6.37	5.47	6.08	6.02	7.44	9.04	4.35	7.50
pk	73.20	40.30	75.29	97.97	3.93	3.51	8.35	8.73	5.26	5.37	8.72	11.09	8.91	8.39
sc	73.95	38.03	3.85	96.25	3.00	2.18	5.99	11.39	3.72	4.06	5.74	5.77	11.83	5.19

* Taken over b1, b2 and b3. (b4 is resampled)

Table A .3 Sample set characteristics (*Bhopal*): train80

Class	avb1	avb2	avb3	avb4	sd1	sd2	sd3	sd4	avsd	cv1	cv2	cv3	cv4	avcv
wl	33.21	15.74	15.73	12.71	0.85	0.61	0.57	1.40	0.86	2.57	3.88	3.64	11.00	5.27
w2	37.99	19.90	20.25	12.34	0.61	0.56	0.65	0.55	0.59	1.59	2.84	3.19	4.46	3.02
wl	34.47	16.89	18.90	27.99	0.55	0.83	0.95	2.28	1.15	1.60	4.90	5.02	8.13	4.91
du	48.03	28.17	41.10	43.15	1.40	1.04	1.65	2.61	1.68	2.91	3.69	4.02	6.06	4.17
sc	37.50	19.98	23.80	38.05	0.62	0.32	1.21	3.94	1.52	1.64	1.59	5.07	10.35	4.66
br	44.03	24.56	34.00	35.29	1.48	1.24	2.35	2.25	1.83	3.37	5.05	6.91	6.38	5.43
dv	43.10	23.05	30.95	28.79	1.40	1.07	1.72	1.34	1.38	3.25	4.63	5.56	4.65	4.52
sv	42.24	25.01	39.92	41.91	0.75	0.44	1.08	1.48	0.93	1.78	1.74	2.70	3.52	2.43
mu	36.21	19.26	20.84	51.08	0.96	0.79	1.95	6.32	2.51	2.66	4.11	9.37	12.37	7.13

Table A .4 Sample set characteristics (*Bhopal*): test88

Class	avb1	avb2	avb3	avb4	sd1	sd2	sd3	sd4	avsd*	cv1	cv2	cv3	cv4	avcv*
wl	33.97	16.48	16.05	11.76	0.76	0.52	0.21	0.50	0.50	2.25	3.18	1.31	4.27	2.75
w2	37.95	19.78	20.11	12.19	0.57	0.53	0.32	0.40	0.45	1.49	2.70	1.59	3.26	2.26
wl	34.35	17.06	19.31	27.82	0.50	0.76	0.78	2.18	1.06	1.47	4.48	4.03	7.83	4.45
du	42.16	22.74	30.68	30.25	1.21	0.92	1.56	1.72	1.35	2.87	4.03	5.08	5.70	4.42
sc	41.08	23.18	35.01	38.01	0.94	0.93	1.77	1.71	1.34	2.28	4.01	5.06	4.49	3.96
br	48.26	27.85	40.41	40.25	1.72	1.52	2.14	2.17	1.89	3.56	5.46	5.30	5.38	4.93
dv	34.85	17.73	19.07	44.81	0.60	0.50	1.09	5.62	1.95	1.71	2.80	5.72	12.55	5.70
sv	36.93	20.00	22.34	40.64	0.37	0.00	1.41	6.78	2.14	0.99	0.00	6.33	16.68	6.00
mu	43.26	23.72	32.09	35.83	1.24	1.01	1.61	2.27	1.53	2.86	4.24	5.01	6.34	4.61

Appendix B

List of various C programs

het.c For the calculation of various statistical measures for the sample sets.

win.c For generation of window file from the band image.

trainpat.c For the generation of *JavaNNS* training file when using only spectral values.

citypat.c For the generation of full pattern set files when using only spectral values.

trainpat2.c For the generation of *JavaNNS* training file when using combined spectral and texture values.

citypat2.c For the generation of full pattern set files when using using combined spectral and texture values.

result.c For generating classified images and error matrices from *JavaNNS* result files.

stats.c For the calculation of various statistical properties from the error matrices.

GLDH.c[†] For the calculation of *asm* texture band from the spectral band.

[†] Developed by Maruthi Ram Ponnapalli, M.Tech. (Geoinformatics), 2001, IIT Kanpur.

Appendix C (Plates)

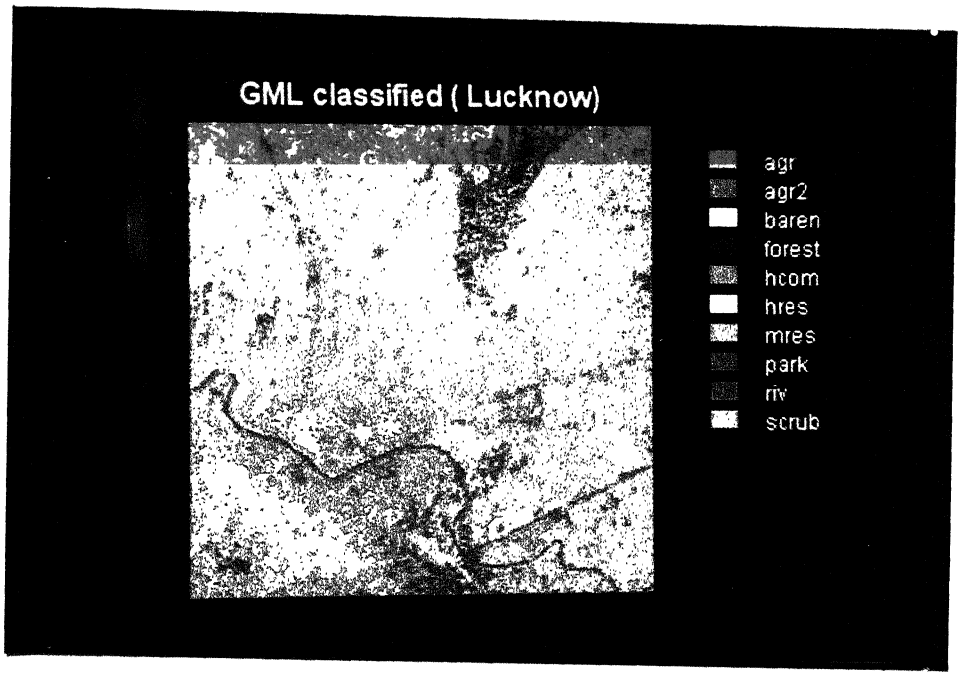


Plate C1.1 GML classified image of *Lucknow*.

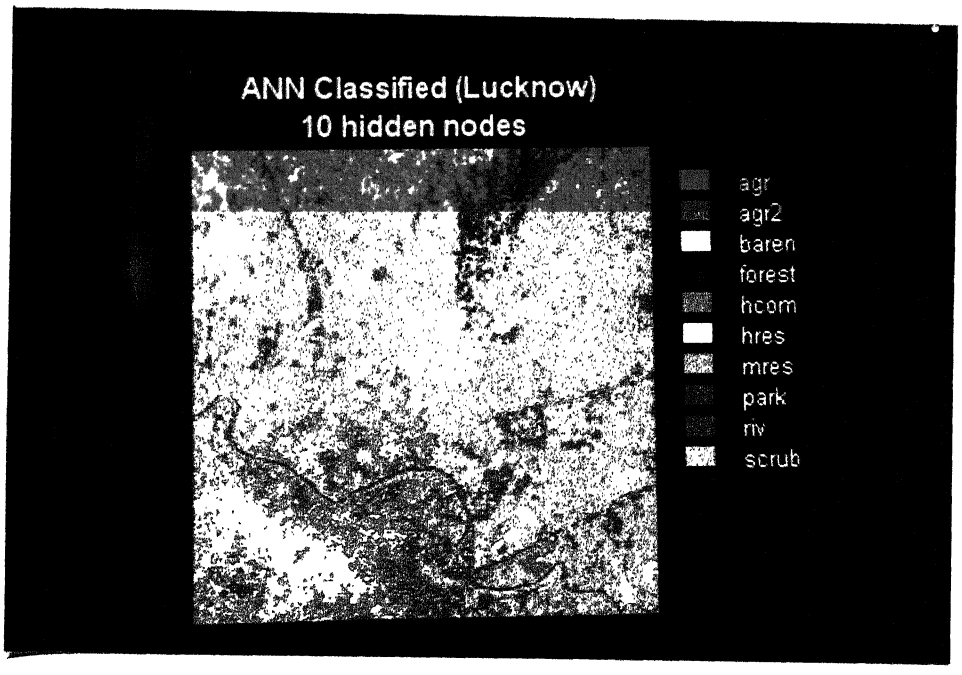


Plate C1.2 ANN classified image of *Lucknow* using 10 hidden nodes.

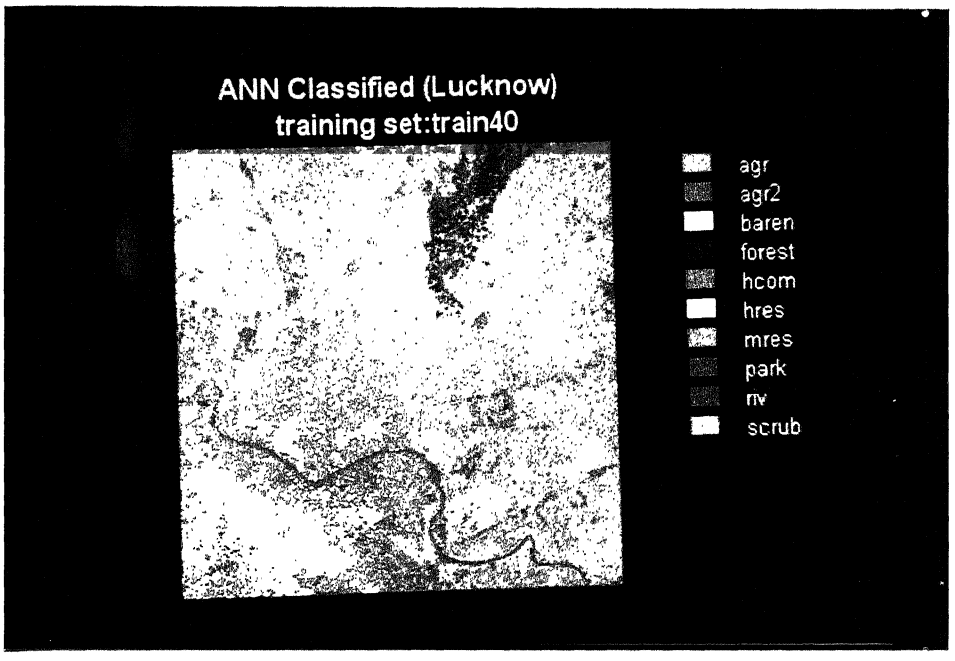


Plate C1.3 ANN classified image of *Lucknow* using *train40* sample set.

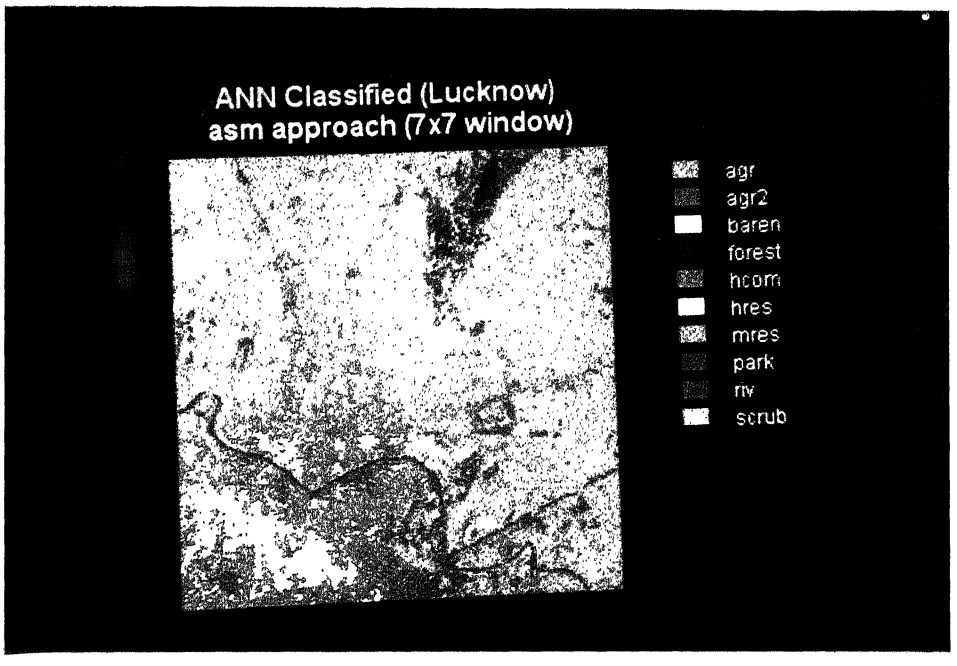


Plate C1.4 MLC classified image of *Lucknow* using *asm* (7 x 7) texture approach.

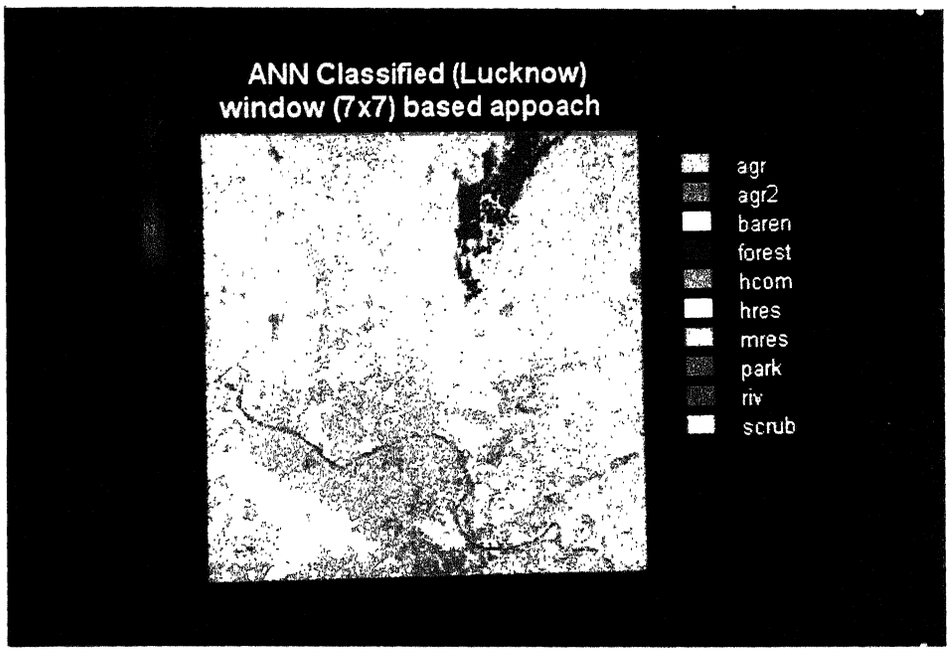


Plate C1.5 ANN classified image of *Lucknow* using window (7 x 7) based texture approach.

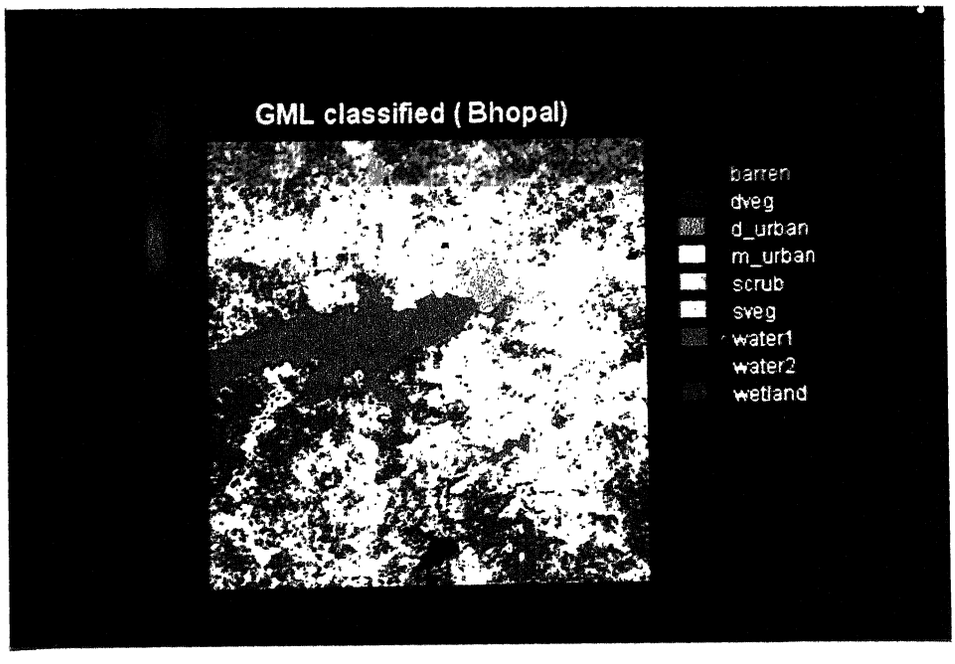


Plate C2.1 GML classified image of *Bhopal.*

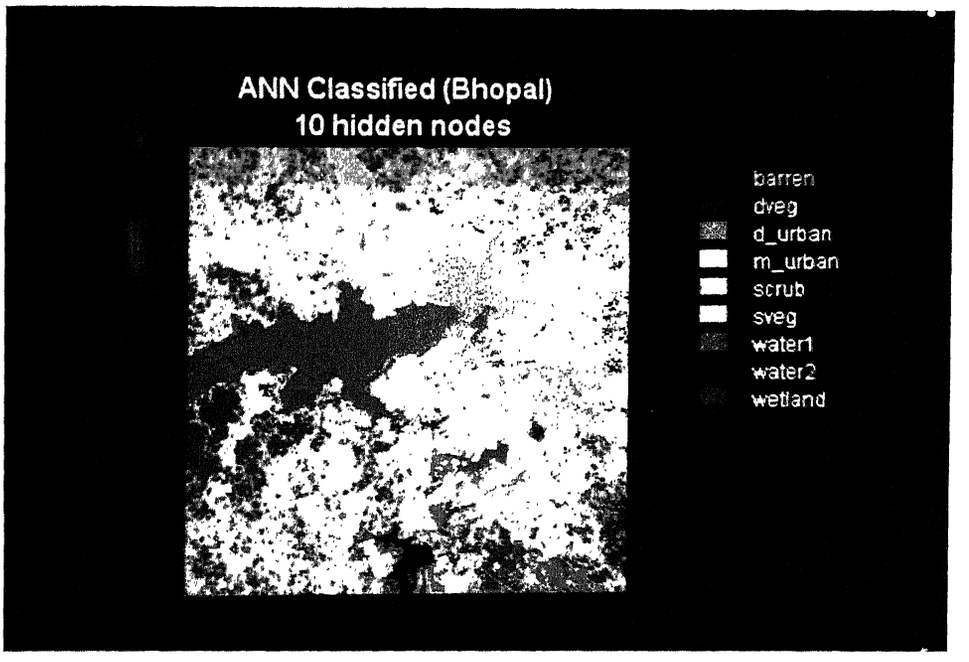


Plate C2.2 ANN classified image of *Bhopal* using 10 hidden nodes

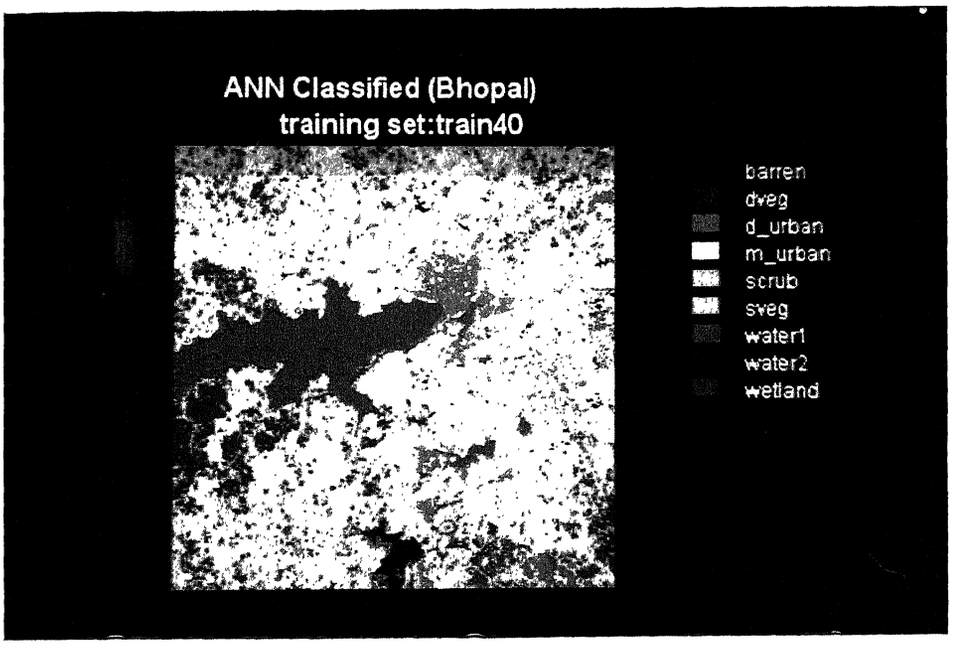


Plate C2.3 ANN classified image of *Bhopal* using *train40* sample set.

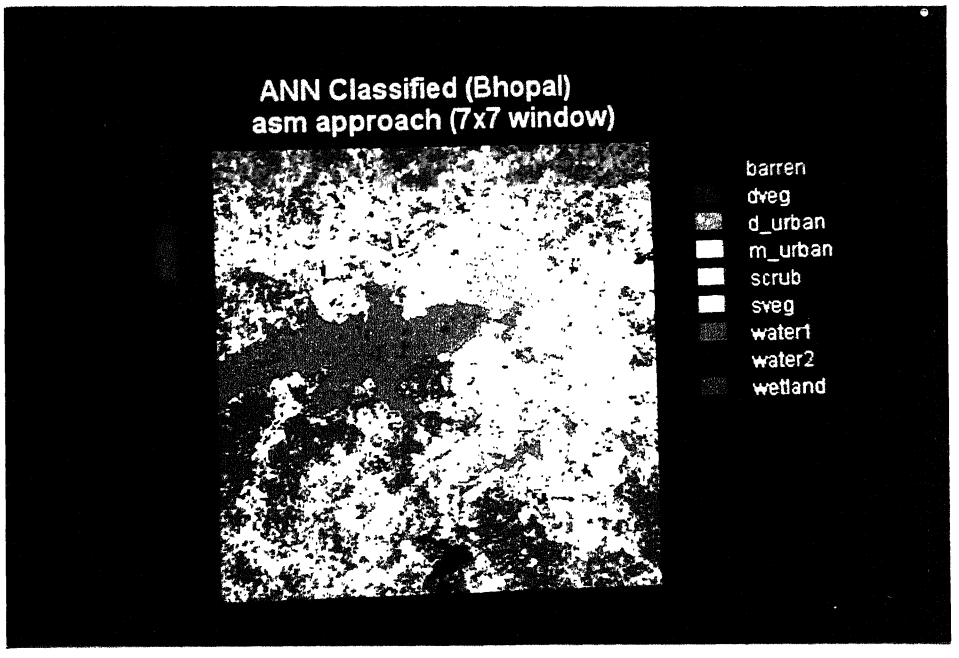


Plate C2.4 GML classified image of *Bhopal* using *asm* (7 x 7) texture approach

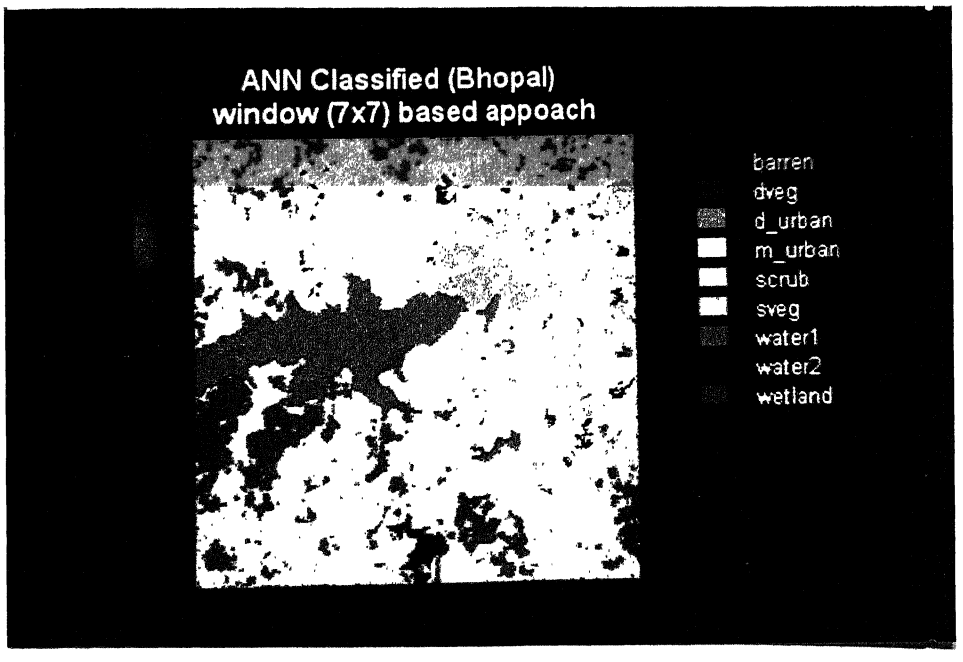


Plate C2.5 ANN classified image of *Bhopal* using window (7 x 7) based texture approach.